

(12) **EUROPEAN PATENT APPLICATION**

(43) Date of publication:
17.11.1999 Bulletin 1999/46

(51) Int Cl.⁶: **G06F 11/34, H04L 12/26**

(21) Application number: **99303188.9**

(22) Date of filing: **23.04.1999**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU
MC NL PT SE
 Designated Extension States:
AL LT LV MK RO SI

(72) Inventors:
 • **Reps, Steven M.**
Bethel, CT 06801 (US)
 • **Luzzi, Joseph**
Ridgefield, CT 06877 (US)
 • **Vedati, Keshavprasad**
Danbury, CT06811 (US)

(30) Priority: **11.05.1998 US 75629**

(71) Applicant: **INTERNATIONAL BUSINESS**
MACHINES CORPORATION
Armonk, NY 10504 (US)

(74) Representative: **Ling, Christopher John**
IBM United Kingdom Limited,
Intellectual Property Department,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) **Client-based application availability and response monitoring and reporting for distributed computing environments**

(57) In a distributed computing environment, a method, system and program product for monitoring, from a client computer system the performance of an application program residing on a server computer system. A probe program residing at the client computer generates requests for the services of the application program and records transaction records based upon service responses therefrom. The requests and transaction record generation is controlled by a set of probe configuration information at the client computer. Transaction records are provided to a central repository whereat statistical information is pre-processed and inserted into statistics tables. A display system enables a computer user to interactively request and view a plurality of displays of data sets of the monitoring data. Each data set includes data elements which may be interactively indicated by the viewer to cause the retrieval and display of related data sets having data elements associated with those in the original display.

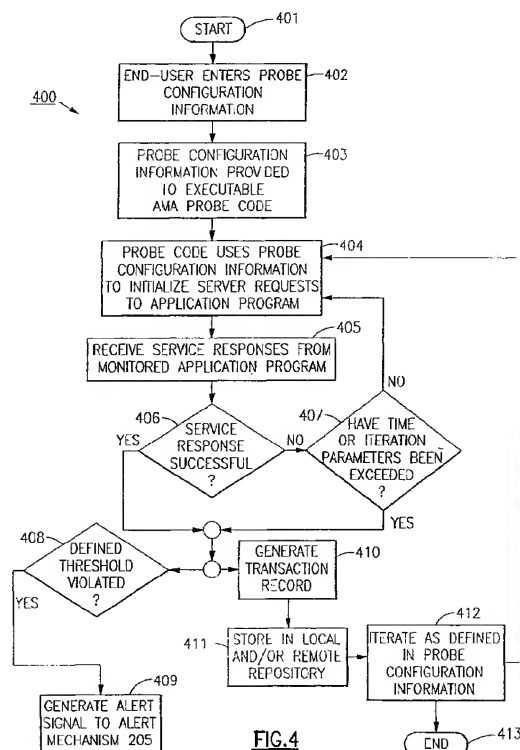


FIG. 4

Description**Field of the Invention**

5 [0001] The present invention is generally related to the field of network system service, and particularly to an end-user based application availability and response monitoring and alerting system, method and program product. More specifically, the present invention enables the monitoring of availability and response time or other desired performance metrics of an application program from the perspective of an end-user utilizing the application program over a distributed computing network. Additionally, the invention provides a readily accessible reporting system for dynamically communicating the real-time results of the application program monitoring. The monitoring system is implemented such that it is not platform specific, does not impact the performance of the monitored applications, and permits easy maintenance. The reporting system enables real-time analysis of the monitoring results, via graphical display, at a variety of levels of granularity available to any user of the network from a central repository thereon. Additionally, the invention includes facilities for establishing performance thresholds for the application program and for determining when the established thresholds have been violated which may indicate performance aberrations of the program. The invention further provides alert signals indicative of the threshold violation to a service or support entity so as to quickly provide the appropriate services to the poorly functioning application.

Background of the Invention

20 [0002] The pervasive current trend in data processing system design is toward utilization of a distributed computing environment wherein an end-user accesses application programs and data over one or more interconnected networks each including multiple interconnected computers. In a typical distributed computing environment, the desktop computers or network computers used by the end-user community, are connected as clients over local area networks (LANs) to a server, which in turn may connect to other such servers locally or remotely. For example, a business enterprise may maintain several interconnected LANs at each of its geographically separate offices. LAN servers at a given office are each interconnected with one another and are further interconnected over wide area networks (WANs) to the servers in the networks of the remote offices.

30 [0003] Businesses have increasingly adopted this computing model in order to allay the cost of operating, maintaining and upgrading separate isolated "piece-part" computing systems. The interconnected networks characterizing this distributed computing model facilitate the prioritization of applications and data, with mission-critical applications and data residing on high-end, high-bandwidth servers, and less important applications and data assigned to correspondingly lower-end servers. In addition, such a highly distributed processing model will typically incorporate features which ensure that the system will continue to function properly and will be continuously available notwithstanding the failure or maintenance of a single or even multiple servers.

35 [0004] Implementation of such a complex, distributed computing model, while offering numerous advantages for its users, presents correspondingly complex network management problems for its network administrators. Heterogeneous operating systems may be implemented in the interconnected networks. Different applications may be running on separate servers as well as different versions or releases of the same application. Failures occurring over localized or distributed portions of the network are not uniformly reported and accordingly corrective actions may be substantially delayed.

40 [0005] In many instances an information technology services (IT) organization either within or outside of the enterprise is charged with the responsibility for managing the distributed computing environment. Typically, a service level agreement (SLA) with such an agency specifies an expected level of application availability and response time for the users of such a network. Adherence to these expected baseline levels is required to fulfil contractual obligations and the failure to achieve these baselines may directly result in the loss of a customer's business. Accordingly, an application monitoring system which provides real-time data regarding application availability and response time would be an invaluable asset to such an organization.

50 [0006] A number of network management tools have been developed to assist the network manager in monitoring the performance of a distributed computing system. For example, the product known as System Performance Monitor/2 available from International Business Machines Corporation (hereinafter "IBM"), provides a graphical interface for depicting the performance of various hardware resources in a processing system, however this product does not indicate the availability and response of a software application to an end-user, and does not permit in depth analysis of the results of the monitoring data. The IBM Netfinity Manager software provides network monitoring of server resources as well as operating system resources at the client level, however it also subsists at the server level and does not monitor client-based access to application programs (Netfinity is a trademark of IBM Corporation). Accordingly, it does not provide the IT professional with information needed to assess whether the aforementioned baseline levels, many of which are specified from the perspective of an end-user or client of the network, are being achieved.

[0007] A number of passive monitoring systems exist for gathering available data from servers and/or clients in a distributed computing system.

[0008] For example, U.S. patent no. 4,858,152 to Estes for "Operator Access To Monitoring Applications" teaches a microcomputer-based monitoring system for concurrently monitoring a plurality of host applications running on a mainframe computer, for summarizing the monitored information and for graphically displaying the information on the display screen of a microcomputer system as well as to provide an alarm mechanism for indicating the attainment of user-defined thresholds. The Multiple System Application Monitor (MSAM) taught by Estes receives existing summarized information from the host machine and reduces the information to an accurate picture of the applications running on the host.

[0009] Likewise, the U.S. patent no. 5,483,468 to Chen et al. for "System and Method For Concurrent Recording And Displaying Of System Performance Data" teaches a performance monitoring tool for interactive selection of performance statistics across a network. The tool incorporates a data supplier daemon which runs on a server to store statistical information which is selectively supplied to a data consumer program which in turn negotiates the reporting of the desired statistics. One advantage offered by the Chen et al. patent is that the data consumer program need not include any prior information regarding the statistics maintained by the data supplier daemon. The Chen et al. patent provides a mechanism for capturing system data and recording the data for subsequent play-back.

[0010] The aforementioned patents, while offering valuable information to a network manager, do not, by themselves, test application availability or response times, but rather they depend upon data being generated by other parts of the system. In the case of Estes, the information is already available at the host for provision to the microcomputer, and in Chen et al., the system statistical data is captured at the server and provided to the data collector. Thus, in both cases these monitoring tools do not generate relevant client-based availability information and are constrained to collecting and reporting pre-existing information on system performance. If no relevant data on application availability and response time from a client's perspective is previously available for these tools, they will not satisfy the objectives of the data manager.

[0011] Several monitoring systems disclose mechanisms for independently generating information indicative of the status of the distributed computing system and collecting and reporting the generated information.

[0012] U.S. patent no. 5,621,663 to Skagerling for "Method and System For Monitoring A Computer System" teaches a system for monitoring and changing the operation of a computer network by modifying an application program to include an event report generator which communicates the occurrence of monitored events to an event processing machine in accordance with a flexible rule base in the event processing machine which associates the occurrence of a particular event with a predetermined action. The event report generator is implemented in the application programs running in the system to report on pre-determined events occurring during the execution thereof.

[0013] U.S. patent no. 5,655,081 to Bonnell et al. for "System For Monitoring And Managing Computer Resources And Applications Across A Distributed Computing Environment Using An Intelligent Autonomous Agent Architecture" teaches a system for managing applications and other server resources wherein an agent is installed in each of the server computers of the network. The installed agents carry out the interrogation functions for identifying which system they reside on, what resources are available and for monitoring aspects of resources and applications present on the server. The agents communicate with manager software systems on the network to enable a continuously updated display depicting all resources and applications present throughout the network and the current state thereof.

[0014] U.S. patent no. 5,675,798 to Chang for "System And Method For Selectively And Contemporaneously Monitoring Processes In A Multiprocessing Server" teaches a monitoring system wherein information regarding the status of each client's application program, as it is reflected by a server process, is acquired and made available to the network administrator. The server process monitor program provides information to the network administrator at the granularity level of each client's process within the client-server network.

[0015] In each of the foregoing examples, the monitoring system requires an intrusive monitor or probe installed at the server level either in the application program running on the server as in Skagerling, or running on the server in a supervisory mode to collect information from monitored applications running thereon. In either case, the results of the probe are not instructive as to the experience of the client since the information is being generated and gathered on the server side of the network rather than the client side. Moreover, the addition of this monitoring code to servers running in the network creates the same maintenance problems as, and may simply be thought of, as adding yet another application to each of the servers. Furthermore, the execution of these monitoring programs may substantially degrade the performance of their host servers, and in turn the networks that they serve, with the dichotomous result that in the name of efficiently managing the network the very tool being used to achieve that objective creates an inefficient network.

[0016] From the foregoing it can be seen that a new application monitoring system which generates availability and response time information or any other desired application program metrics from the perspective of a client would be of great value to a network administrator. The system should be designed to be implemented as a probe at any point within a complex distributed computing environment at which a client computer system may be coupled, and the function

of the probe should have negligible impact on the performance of the network. The system should be customizable to provide real time alert signals alerting a recipient of the traversal of user-defined thresholds such as a maximum tolerable response time or minimum availability of a monitored application program.

[0017] The monitoring system should provide dynamic reports on, for example, application program availability and response time, which can be tailored by the observer to display in graphical or tabular form the real-time and archived monitoring information relevant to the particular observer. The reports should be displayed in such a manner that the viewer may display either via a graph or table or otherwise data relating to the performance of many servers and/or applications and should provide an interactive facility for enabling the viewer to "drill-down" to view data on specific servers or applications and/or to drill up therefrom to a broader view of the performance data.

[0018] The performance report should be readily available to anyone with any type of access to the network and the data therein should reside on a central repository on the network which includes relevant pre-processed statistical information related to the stored data. Access to this information should be provided for persons including the network administrator, help-desk, and end users of the network applications, via wired or wireless connections to the network.

[0019] Finally, the system should be easily implemented and maintained so as to serve as an aid rather than a further burden to the network manager.

Disclosure of the Invention

[0020] The foregoing problems and shortcomings of the prior art are overcome and further advantageous features are provided by the present invention wherein a method, apparatus and program product for implementing a client-based application program monitor in a distributed computing environment is described. Advantageously, the present invention presents techniques for monitoring, reporting and for generating performance-based alert signals, based upon the performance of application services from an application program residing on a server computer coupled to a client computer system in a distributed computing environment.

[0021] In an embodiment of the invention, a computer network includes a server computer having an application program which provides application services to a coupled client computer system wherein the client computer system records information related to the performance of the services of the application program via an application probe software residing on the client computer.

[0022] A set of parameters or probe configuration information are established at the client computer system for use in recording the performance of the application program. The client computer probe is configured in accordance with these parameters to send service requests to the server computer to request the performance of the application services by the application program.

[0023] Correspondingly, the server computer generates a service response which may be an indication that the request is being serviced (a successful response), or which may indicate that the request was rejected (an unsuccessful response), or alternatively, the response may in fact equate to no response from the server system for a pre-defined timeout period (an unsuccessful response).

[0024] Based upon the response that has been received, a transaction record including information related to the performance of the application services by the application program is generated at the client computer.

[0025] Finally, the request and response cycle is repeated in accordance with the parameters supplied to the client computer system.

[0026] In an embodiment of the invention these parameters may include such information as the name of the application program, the address of the server system, the frequency of access to the server computer, the schedule of availability of the application program and threshold information such as minimum availability levels and maximum response time levels as well as the duration of the time out interval for non-responding application programs and whether and how often to retry requests resulting in unsuccessful responses. Other illustrative probe configuration information may include the designations of remote centralized repositories for forwarding the generated transaction records, as well as indications as to whether to discontinue the application monitoring during scheduled periods in which the application program is not available or alternatively to otherwise indicate in the generated transaction records these scheduled periods.

[0027] In a further embodiment of the invention the exceeding of such defined thresholds will cause the client computer to generate and alert signal which may be received and acted upon by service entities to provide problem remediation for the associated application program.

[0028] In an embodiment of the invention the service request is a request to provide a file to the client computer. In a preferred embodiment wherein the service request is a request to perform services for which no client authentication is required.

[0029] In yet a further embodiment of the invention the client computer includes a timer mechanism for determining the response time of the application program as defined by the time elapsed between the service request and the service response. In a preferred embodiment the transaction record information for a generated transaction record

includes whether the application program successfully responded to the request (i.e., whether the application program was available) and the response time of the application program.

[0030] In accordance with a related aspect of the invention, a central repository is provided for receiving the generated transaction records from any of the probes implemented at any of the client computers in the computer network.

[0031] A determination is made based upon the probe configuration as to whether a particular transaction record is to be forwarded to the central repository, and if so, the transaction record is loaded, via a database loading module into a raw data table in the repository. The data in the raw data table is periodically processed to generate statistics which are determined to be of general interest to users of the network. The data required to generate such statistics is extracted at defined intervals from the raw data table by a statistical processing module which in turn generates the desired statistical data and causes the statistical data to be stored in a statistics table in the central repository. In a preferred embodiment the statistical data may include the maximum, minimum and median response times for an application program during a defined service interval as well as a percentage of time that the application was available during that interval.

[0032] In yet another related aspect of the invention means are provided for interactively and sequentially displaying a series of data sets on a viewer's computer. The sequence of data sets are preferably graphs displaying the availability and response times for an application program from one or more client-based probes in the network over a variable period of time such as a month. The data sets comprise data elements corresponding to, for example, the performance of an application program at a set of client-based probes for one day of the month shown in the displayed data set. The data elements are dynamically linked to other data sets each having their own set of data elements related to the dynamically linked data elements from the first set and each of these data elements may be further dynamically linked to other such data sets. The dynamic links are preferably capable of linking to data elements in a second data set and back to the data elements in the first data set from the data elements in the second data set.

[0033] The dynamically linked data elements are responsive to interactions from the viewer such as a mouse-click or-voice commands which cause the viewer's computer to request the related data sets from the repository. The request causes a software-based bridge element to retrieve the requested data from the repository and to provide the data sets to the viewer's computer whereat the related data set having the associated data elements is in turn displayed.

[0034] In an embodiment of the invention, the graphical display may be converted to a tabular representation via a viewer interaction with the graphical display.

[0035] In yet another embodiment, the data elements of the first displayed data sets represent a consolidation of each of the associated data elements within the one or more dynamic linked related data sets. For example, in a preferred embodiment of the invention a data element in the first data set which represents, for example, the response times for a number of different probes monitoring an application program on different servers for a particular day of a designated month, may be dynamically linked to a second data set comprising data elements representing the response times for the each one of the set of probes monitoring the application program on the different servers, for the particular day of the designated month. Moreover, the data elements in this second data set may be dynamically linked to further data sets having further related data elements therein.

[0036] In this manner the viewer, by actuating the dynamic links, may "drill-down" through the monitoring data from a broad view of the application program performance over a period of time on a number of servers, to a very specific view of the performance of a particular application program at a particular client (on a particular sever) at a specific time, and again "drill-up" from that specific view to the same or a different broader view.

Brief Description of the Drawings

[0037] The invention will now be described, by way of example only, with reference to the accompanying drawings in which:

FIG. 1 illustrates a typical network computer environment in which the present invention may be implemented;

FIG. 2 depicts a simple, single probe implementation of the present invention for monitoring the performance of a Lotus Notes application program on a server computer at a client computer and for recording and storing monitored performance data for presentation via a graphical user interface front-end as well as signalling support personnel via alerts if a pre-determined performance threshold is violated;

FIG. 3 more specifically depicts a sequence of operations included in the single probe implementation including the provision of an interface for probe configuration information, the interaction between the probe code and an application program on a server computer, the generation and storage of transaction records and the generation of alert signals;

FIG. 4 includes a flow diagram for illustrating the iterative service request and service response interaction between the AMA probe on the client computer and the application program on the server computer and the generation and storage of transaction records;

FIG. 5 depicts the storage of a probe-generated transaction record in a central repository as well as the scheduled statistical processing of the transaction records into statistical data loaded into the statistics table in the central repository;

FIG. 6 presents a block diagram illustrating the function of the web server computer in providing, upon request, applets for the graphical user interface to a web browser on a viewer's computer and the fulfilment of requests generated by the viewer by interacting with the graphical interface on the viewer's computer, the request being handled via the software bridge element for accessing the data stored in the central repository on the database server computer;

FIGs. 7 and 7a illustrate a display screen for a front-end graphical user interface for the viewer's computer showing scalable measurements of application availability and response time for a Lotus Notes application program running on all listed servers for the Poughkeepsie site for the month of February 1998;

FIG. 8 depicts the data illustrated in FIGs. 7 and 7a in a tabular form which is dynamically linked thereto;

FIG. 9 illustrates availability and response time data for each of the Poughkeepsie site servers running the Lotus Notes application program on February 2, 1998;

FIGs. 10 and 10a present two versions of scaled graphical displays on the viewer's computer for availability and response time data for February 1998 actuated by a viewer by implementing the dynamic link in FIG. 9 for server D01ML010 running the Lotus Notes application program;

FIGs. 11 and 11a present two versions of scaled graphical displays on a viewer's computer for availability and response time data for February 2, 1998 actuated by the viewer by implementing the dynamic link in FIGs. 10 and 10a for server D01ML010 running the Lotus Notes application program;

FIG. 12 depicts the data illustrated in FIGs. 11 and 11a in a tabular form which is dynamically linked thereto;

FIG. 13 presents a flow diagram illustrating the steps undertaken in response to a viewer-implemented indication to actuate the dynamic links presented in the foregoing displays in FIGs. 7-12;

FIG. 14 displays an inverted tree graph illustrating the inter-relationships of the dynamically linked displays typified by those illustrated in FIGs. 7-12; and

FIG. 15 more particularly illustrates features of the alerting function of the present invention.

Detailed Description of the Invention

[0038] The following description will present a preferred embodiment of the present invention wherein a system, method and program product for monitoring, reporting and providing alerts based upon the response time and session availability of a Lotus Notes (Notes is a trademark of Lotus Development Corporation) application program running on a server computer wherefrom the services of the application program are provided to a client computer system coupled via a distributed computing network to the server computer system is provided. Notwithstanding the specific nature of the ensuing description, it will be readily understood by those skilled in the art that such inventive monitoring and alerting techniques as described herein may be employed to assess any desired performance characteristics of any application running in a distributed computing environment from the vantage point of a network resource (i.e., a client of the server) requesting and receiving the services of the application provided from another remote resource (i.e., the server) within the distributed computing environment. For example, and without limitation, such an inventive system, method and program product as described herein may be readily incorporated into a corporate intranet or extranet to assess the availability and response times of hypertext markup language (HTML) based applications running on a server computer and accessed by a client computer via hypertext transfer protocol (HTTP) over an intranet or over the Internet.

[0039] In order to better appreciate the advantageous features of the present invention, it will be instructive to first

consider a typical distributed computing environment in which the present invention may be implemented. Accordingly, Fig. 1 depicts a typical distributed computing environment 100 which may be characteristic of the computing environment of a small to mid-sized enterprise, and which can be expanded upon to include broader networks such as a corporate network or intranet for a large business enterprise or the Internet. In such a distributed computing environment 100 it is likely that there will be one or more networks 101, 102 of varying size, each including a set of server computers 104a, 104b which may provide services and information to a set of client computers also coupled to the network 103 (one of the exemplary client computers may execute the monitoring code in accordance with the present invention). These smaller networks 101, 102, which may for example represent local area networks for a regional office of an enterprise, may in turn be coupled to one or more larger networks 103. The larger networks 103 would typically couple a set of such smaller networks 101, 102 as well as other server computers 104c, and correspondingly enable, for example, client or server computers on any of the smaller regional local networks 101, 102 to couple to server or client computers on the other small local network 102, 101 or on the larger network 103.

[0040] In the course of the ensuing discussion it will be understood that references to server computers correspond to a class of computers which are typically stand-alone computer systems which have been adapted (i.e., programmed) for one primary purpose, that of providing services to individual network users at client computers. Likewise, a client computer may be a stand-alone computer system or any other type of data processing system that has been adapted to interact with server computer systems over a network, including without limitation network computers (NCs). Client computer systems generally refer to any computer system adapted for the use of an individual rather than for use by other computer systems.

[0041] It will be further understood herein that the exemplary distributed computing structure or network 100 depicted in Fig. 1 is merely an illustration of a typical distributed computing environment in which the present invention may be advantageously implemented. A virtually unlimited number of variations of this same basic computing environment including the expansion of the distributed computing environment to the widely distributed computing environment of the Internet will provide a suitable platform for practising the present invention.

[0042] Turning now to our invention in greater detail, reference may be had to Fig. 2 wherein is illustrated a simplified computer network 200 including an application monitoring and alerting (AMA) probe 201 designed in accordance with the principals of the present invention. From a cursory inspection of the network 200 it can be seen that the AMA probe 201 is embodied in a client computer (for example computer system 106 in Fig. 1) which is coupled to a server computer 202. The server computer 202 includes an application program 203, the performance of which will be assessed by the monitoring and alerting activities of the AMA probe 201.

[0043] In operation, the AMA probe 201 establishes a session with the server computer 202 by requesting the services of the application program 203 operating thereon. Session establishment is actuated by a service request 210 sent from the AMA probe 201 over a network link 206 to the server computer 202. Correspondingly, the server computer's application program 203 provides a service response 211 over a network link 206 back to the requesting AMA probe 201.

[0044] It is particularly noteworthy that the foregoing sequence of transactions is precisely the same sequence of transactions undertaken by a customer of the computer network 100 seeking the services of the application program 203 on the server computer 202 at his/her client computer 106. In essence, by monitoring an application program and providing alerts based upon the aforementioned sequence from the viewpoint of the AMA probe 201, the present invention achieves a realistic picture of the performance of the application program 203 from the vantage point of a customer using the application program over the distributed computing system 100 at client computer 106. By implementing the AMA probe 201 in this manner it is possible to gather real-time information based upon an "end-user's" experience of a client-server based application program 203. The AMA probe 201, in accordance with the preferred embodiment of the invention, quantifies the availability and the response time or other desired performance aspects of such an application program in relation to pre-defined and often contractually enforced performance criteria for the network 100. Consequently, based upon the results obtained from such a probe it is a relatively simple matter to establish such performance criteria for a given application program over a given network and correspondingly to identify instances wherein an application program on the network fails to meet the established criteria. Through the facilities of the present invention, a network manager may establish and monitor adherence to SLA objectives.

[0045] Turning to a more detailed consideration of Fig. 2, it is noted that the AMA probe 201 may receive a number of different types of service responses from the server computer 202. For example, if the application program 203 on the server computer 202 properly responds to the service request, the AMA probe 201 will receive an indication of a successfully completed request i.e., a successful service response, from the server computer 202. Alternatively, if the server computer 202 is unavailable to respond to the service request 210, the request will timeout after a predetermined period and the AMA probe 201, based upon receiving back no response for the time out period, will record that the server computer was not available. This can be viewed as an unsuccessful service response 211. Finally, if the server computer 202 rejects the service request 210 the AMA probe will again record the transaction as an unsuccessful service response 211. A rejected service request 210 may correspond to a variety of different situations, such as,

wherein the client is not authorized to access the particular server or application program 203 thereon, or wherein the application program 203 has been temporarily taken "off-line" for maintenance purposes or if the application program 203 is functioning improperly for any number of reasons.

[0046] Whether it is successful or unsuccessful, the service response 211 from the application program 203 on the server computer 202 (including the determination of a no-response time-out) is received at the AMA probe 201, which then records the results of the transaction in a database repository 204.

[0047] The database repository 204 may be local to the AMA probe on the client computer 106 and/or remote from the probe at another point on the network 200 coupled thereto by a network link 207 or in a preferred embodiment, the invention may include both a local and remote data repository 204 and data may be stored locally and forwarded either concurrently or subsequently to a centralized remote repository which collects probe data from a number of probes monitoring different applications at different points on a distributed computing network 100. In the preferred embodiment, the centralized database repository 204 for recording the transaction records from multiple probes 201 on the network 100, is designed to be accessible to any user of the distributed computing network 100 and would ideally provide a front-end graphical user interface (GUI) 212 such as an intranet-based web site accessed via hypertext transfer protocol (HTTP) and providing a page or pages of hypertext code (i.e. hypertext markup language or HTML pages) which would permit anyone within the enterprise to easily access and analyze the data stored therein. Further details of such a centralized reporting mechanism including a graphics-based reporting interface will be subsequently described herein.

[0048] In a further preferred embodiment of the present invention, facilities are provided for accommodating the establishment of a set of performance criteria for the monitored application program 203. In such a preferred embodiment, the AMA probe 201 determines whether any pertinent elements of the completed transaction including the service request 210 from the probe and the corresponding service response 211 from the application program 203 on server computer 202 have exceeded such pre-defined criteria. The pre-defined performance criteria may include such metrics as maximum allowable response time, and/or a maximum number of failed successive attempts to access the services of the application program (i.e., an indicia of session availability).

[0049] A determination that one of these pre-defined performance criteria has been violated prompts the AMA probe 201 to generate an alert signal 208 which is sent to an alerting mechanism 205, which in turn is designed to inform an appropriate support entity of the violation such that problem determination and remediation steps may be quickly implemented.

[0050] To better illustrate the foregoing transaction sequence we will refer to a preferred embodiment of the present invention wherein a Lotus Notes application program 203 is included on the server computer 202. The client computer 106 including the AMA probe 201 further includes configuration facilities for implementing the function of the probe.

[0051] Referring now to Fig. 3 it can be seen that these configuration facilities include a GUI-based front-end 301 which is provided to elicit information pertaining to the function of the probe (i.e., probe configuration information 302) from the party responsible for the implementation of the AMA probe (i.e. from the end-user). In an exemplary embodiment, the probe configuration information 302 will include the name of the server computer 202 including the application program 203 for which monitoring is desired (i.e., the target server name), the network address of the target server and the type of application on the target server to be monitored (for example Lotus Notes). This basic set of information is necessary to establish the initial sequence of network communication operations comprising the aforementioned transaction sequence undertaken by the AMA probe 201 including the generation of a service request from the AMA probe 201 on the client computer 106 to an application program 203 on a server computer 202.

[0052] Further items of probe configuration information 302 which may be elicited in an embodiment of the present invention include access control and authentication information, if required, for accessing secured transactions (i.e., such information as user identification and password), as well as instructions regarding the processing of the information gathered based upon the service responses 211 of the application program 203.

[0053] The information required for processing the service responses 211 from the application program 203 on the server computer 202 may include, without limitation, a storage designation, indicating the repository 204 for storing information regarding the completed transaction (i.e., the transaction records). As previously noted, this repository may be local 305 to the client computer 106 and/or a remote central repository 306 on the network 200 which stores data gathered from multiple probes 201 monitoring multiple application programs 203 on multiple server computers 202. In a preferred embodiment of the invention, storage operations may be undertaken both local 305 to the client computer 106 including the AMA probe 201, and remote on the network 100 at the central repository 306, and the repository designations will be provided for both repository locations 305, 306.

[0054] In still a further embodiment of the invention, probe configuration information 302 would be elicited for determining the sampling frequency for generating the service requests 210 to the application program 203 on the server computer 202. It will be further appreciated that the AMA probe 201 may be instructed to monitor multiple application programs 203 either on the same or on different server computers 202. In such a multiple monitoring implementation, the sampling frequency may be defined locally for each application to be monitored or globally for all or for a portion

of all of such applications 203 monitored by the probe 201 on the client computer 106.

[0055] As previously noted, a preferred embodiment of the invention comprises facilities for generating an alert signal 208 for actuating an alerting mechanism 205 signalling the violation of pre-defined performance criteria. Accordingly, such an embodiment will require the establishment, as a part of the probe configuration information 302, of these pre-defined performance criteria. For example, it would be a simple matter to define a maximum allowable response time for receiving a successful service response from a monitored application program 203. As will be subsequently described, the AMA probe 201 generates a transaction record including the duration of the transaction cycle from service request to service response. If this duration exceeds the pre-established threshold, an alert signal 208 will be generated by the AMA probe 201 and forwarded to the alerting mechanism 205.

[0056] In an embodiment of the invention wherein it is desired to establish an availability threshold for an application program 203 as part of the probe configuration information 302 which when exceeded will trigger the generation of an alert signal 208, the process may be implemented by determining a maximum number of successive unsuccessful service responses received by the AMA probe 201 from the application program 203 on the server computer 202.

[0057] A particular configuration of the AMA probe 201 for determining an availability performance criteria for the application program 203 might entail, for example, the establishment of an availability threshold designed in accordance with the foregoing description, coupled with configuration information instructing the probe 201 to immediately, upon the receipt of an unsuccessful service response 211, re-initiate the service request 210 until either a successful service response 211 is received or the threshold is exceeded and an alert signal 208 is generated. In this manner, it is possible to closely relate the generation of an application availability-based alert signal 208 with the loss of the availability of a particular application program 203, depending of course upon the defined sampling interval 302 for generating the initial service request 210.

[0058] Since any server computer 202 on a distributed computer network 100 will require periodic maintenance which often requires that the server computer 202 is not available to provide application services to client computers 106 on the network, it would be advantageous to account for such known "service outage intervals" so as not to perform application monitoring during these intervals or alternatively to condition the results of monitoring done during the maintenance outage periods by associating these results either with a flag or other indication that these results were obtained during such an outage interval. Moreover, it may simply be the case that the application program 203 is not run continuously but only at certain scheduled times. Accordingly, a further embodiment of the invention would elicit such information as the availability schedule for the application program 203 as part of the probe configuration information 302.

[0059] Each of these aforementioned items is included in the probe configuration information 302 of a preferred embodiment of the invention which are collected by the GUI-based AMA probe interface 301 depicted in Fig. 3 and further illustrated in tabular form in Fig. 3. In accordance with the illustration in Fig. 3, probe configuration information 302 collected by the interface 301 is provided to the executable portion of the AMA probe 303 on the client computer which in turn functions by generating service requests 210 to the identified application program 203 on the target server 202 in accordance with the configuration information 302 provided by the GUI 301.

[0060] Considering once again our exemplary embodiment wherein a Lotus Notes application program 203 on a server computer 202 is to be monitored by the AMA probe 201. Once the probe configuration information 302 has been entered into the graphical interface 301, the AMA probe executable code 303 resident on the client computer system 106 utilizes an application program interface (API) 304 for the application program on the server computer, in the same manner as an end-user client seeking the services of the application program 203 over the network 200. However, the frequency of access, userid, password, etc. with which the probe code 303 requests the service response from the application program interface 304 are dictated by the probe configuration information 302 supplied through the graphical interface 301. In the case of the Lotus Notes application 203 the AMA code 303 utilizes the Notes APIs 304 included in Lotus VIM (Vendor Independent Messaging) API Toolkit.

[0061] In an illustrative embodiment of the invention including the Lotus Notes-based monitor, the transaction between the probe 201 and the application 203 entails a request from the probe to the Lotus Notes database server to open a Lotus Notes database file called README.NSF. This transaction, while not a common user request for a Lotus Notes database, mail or hub server, is indicative of the availability and response time of the application program in general. Such a simplistic service request 210 is ideally suited for monitoring purposes in the present invention, since it is not required to authenticate the requesting session via a password to secure a successful response 211. Furthermore, this particular service request 210 does not necessitate processor-intensive operations at the server computer 202, and accordingly the transaction which is iteratively performed over the computer network 200 does not materially impact the overall performance of the server computer 202 for other client computers 106 on the network 100. It will of course be understood that the choice of the particular transaction to be monitored is to be left to the entity administering the probe and is not in anyway to be limited by the particular description herein.

[0062] The completion of such a client-server transaction comprising the service request 210 and the service response 211, culminates in the generation of a transaction record 311 which is forwarded via the AMA probe code 303

to a storage repository 305, 306. The transaction record 311 includes information regarding the transaction, such as whether the application program 203 has successfully responded to the service request 210, the total duration of the transaction cycle from request to response, the time of day of the service request and any other metric which may be of interest to the entity seeking to monitor the particular application program 203.

[0063] In determining the cycle duration, a timer mechanism 307 is included in the AMA probe code 303. In a preferred embodiment of the invention the timer mechanism 307 may simply be a mechanism which places a time signature on the initial service request 210 from the probe 201 and another time signature on the service response 211 at the probe 201 and which records the difference between these two time signatures in a transaction record 311. In the preferred embodiment including the Lotus Notes monitor, availability and response time are the key metrics for which data is included in the stored transaction record 311. An exemplary transaction record from the AMA probe 201 is illustrated below:

Meas	Monitor	Target	TgtSrv	Srv	Monitor	0=F
Date	Local Time	Server	Type	Loc	Name	1=S
"05/28/1997"	"19:54:40"	"D02DBE01"	"SPM"	"SBY"	"SBY"	"1"

IP Ping	Notes DB	Request
Time	Resp. Time	Intervl
"0.000000"	"1.281000"	"6"

It will be observed in the above-listed transaction record that the particular transaction recorded a successful service response (wherein a "1" is recorded in the field "0=F, 1=S") and the response time was 1.281000 seconds for the transaction.

[0064] In addition to providing transaction records to the repositories 304, 305, an additional feature of the present invention provides the real-time results of the transaction back to the relevant fields of the GUI-based interface 301. In this manner, a user may view the response time and availability information corresponding to each transaction once the AMA probe code 303 has processed the transaction record, if desired, by viewing the GUI-based fields 301.

[0065] As has been previously discussed, the records from the transactions may be stored locally in a repository 305 on the client computer and/or remotely in a centralized repository 306. The instructions for such storage operations are included in the probe configuration information 302 designated at the graphical interface 301 level. In an alternative embodiment it is possible that the records will not be stored in any repository. This would be the case, for example, wherein only alerting of violations of performance criteria is desired from the probe or wherein only real time transaction information is of interest to the network administrator. Such implementation choices are a function of the desired objectives of the application monitoring project and all are enabled by the present invention.

[0066] Referring again to Fig. 3 it will be further appreciated that upon generating the transaction record, the probe 201 further performs threshold comparison operations required to determine whether any of the interface-based (301) pre-determined performance criteria have been violated. If it is determined that one or more of these criteria have been violated, an alert signal 308 is generated by the probe 303 and forwarded to the alerting mechanism 205 for informing service personnel of the violation incident.

[0067] Fig. 4 summarizes the steps involved in a processing a client-server transaction 400 in accordance with the present invention. The process is initiated at step 401 and proceeds to step 402, where the client computer system including the AMA probe software 201 prompts a network-user via a GUI template 301 or otherwise to enter the probe configuration information 302 to control the function of the probe code 303. Next in step 403, the probe configuration information is provided to the executable portion of the AMA probe code which uses the information to initiate a series of service requests 210 in step 404 to a monitored application program 203 on a target server computer system 202. As previously noted the target server 202 and application program 203 are included in the information provided to the AMA probe code 303 via the template 301 in step 402.

[0068] The service requests 210 generated by the probe elicit service responses 211 which are received by the probe code 303 in step 405. Next in decision box 406 the code 303 determines whether the service response is successful. As previously described, in the case of successful service responses, the probe receives an indication that the service request was received by the application program and is being fulfilled. In contrast, an unsuccessful service response may correspond to either no service response from the application program prior to the expiration of a pre-defined time-out period or alternatively to a service response from the application program indicating that the service request is not being fulfilled. In the event that an unsuccessful service response is received by the probe 303, the probe may respond (depending upon the set-up of the probe via the probe configuration information 302) by immediately retrying

the service request 404, the retry operation, if enabled could comprise a single or multiple retry attempts which may be bounded by either number of retries or a retry time period as is indicated at decision point 407. Upon exceeding the determined retry interval without eliciting a successful service response, a transaction record for an unsuccessful service response is recorded 410.

[0069] In either case, whether the service request produces a successful or unsuccessful service response, a determination may be made as to whether any pre-defined thresholds, which have also been defined in the probe configuration information in step 402, have been violated 408. If it is determined that such thresholds have been violated, an alert signal is generated by the probe 409 and passed to the alerting mechanism 205.

[0070] Concurrently, or subsequent to the threshold determination 408, a transaction record 311 is generated by the AMA probe code 303 in step 410. The transaction record 311, as illustrated above, includes information pertaining to the specific transaction (i.e., the service request and response cycle) that has just completed. The transaction record is next stored in a local and/or central remote repository 305 and/or 306 in step 411. The cycle beginning with the initiation of a successive service request in step 404 is repeated in step 412 if iteration has been instructed and at the frequency defined by the probe configuration information. Finally, the process may be viewed as being bounded by ending block 413.

[0071] In accordance with the foregoing description it can be seen that the AMA probe 201 may be implemented to generate transaction records 311 which are iteratively appended to a repository which may be local 305, or centralized 306. The centralized version of this repository 306 will include a history of application service transaction records 311 for each monitored application program 203 on each server computer 202 being probed in a distributed computing system 100. A further feature of the present invention relates to a method, system and program product for providing this stored information to users of the distributed computing environment 100. Another feature of the present invention relates to a method, system and program product for processing the alerts 308 generated by the AMA probe code 303 and for reporting the alert conditions to support personnel.

[0072] Referring now to Fig. 5 we are presented with an overview of the implementation and organization of the centralized data repository 504 used for storing and providing reports for applications service transaction records 311. More specifically, Fig. 5 illustrates the insertion of real-time transaction record data 311 from the probe code 303 into the centralized data repository 504 and the generation of statistical data 506 based upon the inserted real time data 505.

[0073] In operation, a transaction record 311 is generated by the AMA probe code 303 at the conclusion of a transaction cycle. The record is provided to a database loader module 502 which inserts the real-time transaction record data into the raw data table 505 in the centralized AMA data repository 504. Record insertion from the database loader 502 is actuated upon the receipt of a transaction record 311 from any of the probes 201 running on client computers 106 in the distributed network 100. As such, it can be seen that the raw data table 505 of the repository 504 includes real time data provided at approximately the time that the transaction records 311 are generated by the probe code 303 in addition to any inherent latency associated with passing such data over the network 100.

[0074] As a consequence of the application monitoring activities undertaken by the AMA probes 201, there exist identifiable sets of statistics, based upon the raw data 505, which are known to be of interest to end users of the network 100. These statistic sets include, for example, the percentage of successful service responses 211 per the total number of attempted service requests 210 in a determined interval of time. This pre-determined time interval typically coincides with the SLA between the network administrators and the network customer (for example this interval would typically correspond to a prime-shift work day as defined by the SLA). Alternatively, statistical processing may be actuated upon the receipt of a defined number of transaction records 311 into the raw data table 505, such that upon receiving 250 transaction records into the table the statistical processing for the received 250 records will be initiated. Of course, this numeric threshold may be further divided into the receipt of a number of records corresponding to the monitoring of a specific application program 203 on a specific server 202 etc..

[0075] By processing raw data 505 within the central repository 504 at meaningful intervals, the desired statistical sets of data 506 are made readily available to an end-user who may wish to ascertain some measure of the performance of the application(s) 203 on the network 100. From the vantage point of such an end-user seeking a report on the performance of applications being monitored, the raw data in table 505 is periodically pre-processed into statistical sets of data 506. Since the provision of real-time statistical computation may be a processor-intensive task, this pre-processing of statistical information provides an efficient reporting mechanism to the end-users of the network who are querying this information.

[0076] The pre-processing of the statistical information is enabled via the function of the database reporter 507. At pre-defined intervals the reporter accesses the data inserted into the raw data tables 505 by the database loader 502 and processes the information thereby generating the defined sets of statistical data which it then inserts into statistic tables 506 for access by end-user queries.

[0077] The database reporter 507 is actuated at intervals coinciding with the establishment of a requisite set of raw data for formulating the sets of statistical information. For example, in a preferred embodiment of the invention, a calculation is performed at the conclusion of each business-day prime-shift (i.e., 7:00 AM - 6:30 PM) based upon raw

data which has been iteratively generated at probes 201 and inserted by database loader 502 into the in raw data table 505 of the central repository 504. Alternatively, as previously noted, the calculation may be performed based upon receipt of a defined number of transaction records 311 at the raw data table 505.

[0078] This inserted raw data corresponds to transaction records 311 from AMA probe code 303 which has been monitoring application programs 203 during this interval. At the conclusion of this interval, it is a simple matter to provide the newly inserted raw data to the database reporter 507 which in turn will pre-process the data to provide statistical records on the availability of each probed application program during that interval (i.e., by determining the percentage of successful service responses recorded for each application during the interval). Moreover, the statistical information may additionally include a range of the response times recorded for the pre-processing interval. The pre-processed statistical information is then provided to the statistics table 506 of the central repository 504. In this manner this information is calculated only once and may be quickly provided to an end-user querying the central repository. The types of statistics inserted into the statistics table 506 will be more apparent via subsequent discussions regarding the front-end GUI's provided to a viewer of the central repository 504 as described with respect to Figs. 7-14.

[0079] In an exemplary embodiment and as will be subsequently described via reference to Fig. 6, an end-user may receive this stored information over a web-browser application program by coupling to a web server program 600 within web server computer 508 which may access the statistical information stored in the central repository 504 without the need to process the statistical information on demand.

[0080] Turning now to Fig. 6, the function of the web server computer 508 is further illustrated. The central repository 504 resides on a database server 601. The end-user seeking to ascertain application program performance data on the network, establishes a session 608 with web server program 600 via a web browser application program 602 on a computer system such as client computer system 106 (which may or may not include the AMA probe code 201), which is capable of coupling to the database server 601 and the web server program 600 in web server computer 508. In a preferred embodiment, the web server program 600 stores a software package 603, which may be a Java(tm) class file. The software package 603 is provided upon request to the end-user computer 106. Upon receipt at the end-user computer 106 it is interpreted by the web browser 602 to provide the AMA Java applet 604 which in turn functions to provide a graphical indication of the desired performance data to the end-user over the web browser 602 (Java is a trademark of Sun Microsystems Corporation). When an end-user communicates with the web server program 600 via a uniform resource locator (URL) an authentication process 608 is initiated based upon the information provided from the web browser program 602. Once authenticated, HTML code is provided 608 from the web server program 600 to the web browser 602, after which the Java-based software package 603 is provided 608 to the web browser 602.

[0081] At the end-user computer the received Java package 603 is interpreted by the web browser 602 which creates the AMA Java applet 604. The AMA Java applet 604 serves as the interface between the end-user and the data stored in repository 504. The AMA Java applet 604 includes a GUI (AMA GUI 605). The AMA GUI 605 comprises a set of graphical and tabular objects which are used to assist the end-user in requesting data 609 and analyzing the data subsequently retrieved 609 from the database repository 504 on database server 601. As part of the interpreting of the AMA applets 604, the session 608 established between the end-user computer 602 and the web server program 600 is terminated. Upon termination, the web server 508 need not be utilized to perform computations and communication required for accessing the data stored in the repository 504, accordingly, the web server 508 is available to serve other end-users seeking establish a session therewith.

[0082] The end-user computer 106 next generates requests for data via the AMA GUI 605 from the repository 504 which has the effect of initiating a temporary data link with the database server 601 for receipt of the data stored in the repository 504. The AMA bridge 606 is a software-based interface mechanism which is logically coupled via a continuous communication session 610 to the repository 504. The bridge 606 is given access to a portion of the repository which is intended to be accessed by end-users via web browser 602 running the AMA applets 604. It will be recalled that authentication of the end-user was accomplished during session establishment 608 between the web browser 602 and the web server program 600. As such, there is no need to authenticate an end-user's access to the repository 504 through the bridge 606. The AMA bridge 606 manages the high-speed bidirectional data link 609 between the end-user session 602 and the repository 504 on database server 601 through the AMA applets 604. In this manner the data-intensive graphical presentation materials for graphically representing the probe monitoring results via the AMA GUI 605 are provided as a packet 603 of software locally implemented 604 at the end-user's web browser 602. The comparatively limited data corresponding to the probe monitoring results stored in repository 504 is transferred between 609 the database server 601 and the end-user's computer 106. This data is processed by the AMA applet 604 for display via the AMA GUI 605.

[0083] The temporary user-to-database data link 609 is actuated by the user's interaction with the AMA GUI 605 included in the AMA applet 604 on the user's web browser 602. For example, by clicking on templates 605 displayed locally on the user's web browser 602, the user causes his/her computer 602 to request data stored in the repository 504 on the database server 601. The AMA bridge software 606 serves an interface for the high-speed transfer of the relatively limited data communicated via data link 609 between the end-user web browser 602 and the repository 504

on database server 601. The bridge software 606 is implemented as a daemon continually logically coupled via a communication session 610 to the repository 504, and in an exemplary embodiment may physically reside on the web server 508 or on the database server 601 or on an intermediate computer (not shown). The bridge daemon 606 provides services for the applets running on the end-user web browser 602 accessing the AMA GUI 605. The bridge 606 receives requests from the end-users through the GUI 605, and responds with the appropriate data from the repository 504.

[0084] Using Java Database Connectivity (JDBC) the bridge daemon 606 is continually connected 610 to the repository 504 such that multiple requests to the repository do not require the creation of multiple database processes (as would be the case if another form of access such as via Common Gateway Interface (CGI) were utilized). Accordingly, multiple requests from any number of end-users are managed via a FIFO queue (or otherwise) through the bridge daemon 606. As such the bridge daemon 606 provides an efficient mechanism for connecting the end-user directly to the central repository 504.

[0085] It will be further understood that multiple end-users (n) may each access the web server program 600 at a given time and may each be provided with the Java class files. Accordingly, since the session between the end-users and the web server 508 is terminated upon the provision of the Java packet 603, over the course of time a great number of end-users (m where m may be » n) may locally implement the AMA applet 604 including the AMA GUI 605 and may establish data links 609 to the bridge element 606. The single continuous communication session 610 established between the bridge element and the repository 504 will serially couple the m end-user computer 106 to the data stored in the repository 504 by sending the requested data over the m data links established between the end-user computers and the bridge 606.

[0086] Upon establishing a session 608 with the web server program 600, an end-user indicates via a main menu template provided in web server program 600, which monitored application(s) on which set of servers he/she is interested in. The web server program 600 receives this information and provides 608 the appropriate Java packet 603 required for creating the appropriate AMA applets 604 including the AMA GUI 605 on the end-user's web browser 602. The end-user's interaction with the AMA GUI 605 executing locally within the web browser 602 on his/her computer 106 actuates data link to the bridge daemon 606 which is continually coupled to the repository 504 and which manages the data exchange therebetween.

[0087] From the foregoing descriptions it will be readily apparent that the database repository 504 may typically include voluminous sets of transaction records 311, 505 and statistical data 506 relating to the performance of a plurality of monitored application programs 203 which have been/are presently being monitored by a plurality of AMA probes 201 executing on a plurality of client computers 106 on the distributed computing network 100. It will be further appreciated that the usefulness of this stored data 504 is directly related to the facility with which the data may be analyzed. Accordingly, a further feature of the present invention comprises graphical display facilities for presenting the stored application monitoring information 504 to an end-user in an easily comprehensible fashion.

[0088] In particular, the present invention provides interactive report generation and graphical presentation facilitates enabling a viewer to query and obtain graphical and tabular data corresponding to monitored application program performance on the distributed network 100 at both a high level wherein, for example, the availability and response time of multiple servers running an application program or multiple application programs is displayed on the basis of performance data gathered over a month or a longer time period, and at levels of increasing granularity ranging to the level of a display or table comprising data corresponding to the hourly performance of a specific server running a specific monitored application program. Each of these levels of data in each of the tables or graphs is dynamically linked to successive tables and graphs and to previous tables and graphs having a broader and/or narrower views of the data that is to be analyzed as desired by the viewer. Traversal of these linked tables and graphs provides a user-friendly querying tool for retrieving stored data from the central repository 504 and for displaying the data at different levels of specificity in a manner that is readily user-comprehensible.

[0089] Notwithstanding the fact that the ensuing description will refer to interactions with the presentation system by way of computer mouse operation, it will be understood that within the present invention the viewer is provided with the ability to interact with these graphical or tabular representations by any known computer interface mechanisms, including without limitation pointing devices such as computer mice or track balls, joysticks, touch screen or light pen implementations or by voice recognition interaction with the computer system. The user interaction specifies the successive levels of data to be displayed to the viewer.

[0090] Turning now to Fig. 7 - 12 we are presented with an exemplary sequential set of viewer interactive graphical and tabular presentations which are implemented as a portion of the AMA GUI 605 in a preferred embodiment of the invention.

[0091] The graph 700 in Fig. 7 is generated by when a user fills out the template fields illustrated 707 along the bottom of the graph 700. The fields 707 include the AMA features to be displayed (response time and availability in the illustrated example), the server (or all servers as in the example) running the application program for which monitoring results are to be displayed and the time period for the monitoring (February 1998 in the example).

[0092] By filling in these fields, the user causes the AMA GUI 605 to generate a request to the bridge daemon 606

which in turn queries the repository 504 and returns this information back to the GUI for display in graph 700. It will be understood as we proceed to describe the features of the following dynamically linked displays in Figs 7 - 12, that the interconnecting links or active areas on each table or graph, when actuated by the user to traverse the subsequently described series of graphs and tables, each cause a request for data to pass from the AMA GUI 605 to the bridge daemon 606 and further cause the querying of repository 504 by the bridge daemon 606 and the return of the requested data therefrom to the GUI 605, which in turn assembles the returned data via the AMA applet 604 into a display format such as shown in the graphs and tables of Fig. 7 - 12.

[0093] With the foregoing descriptions in mind we now turn to Fig. 7 which illustrates a graph 700 showing the response time and availability for a Lotus Notes mail application on a set of AIX servers for the month of February 1998 (Lotus Notes (AIX) Mail). The X-axis of the graph 700 corresponds to the days of the month. The Y-axis of the graph indicates a first scale 701 corresponding to an indicia of percent availability per day of the month of the monitored Notes application, and a second scale 702 corresponding to an indicia of the response time (in seconds) for the monitored application transaction. The ranges for response time and availability can be scaled up or down as desired by clicking on the "+" and "-" representations shown for each scale 702a and 701a respectively. For example, Fig. 7a shows the same graph 700a having an expanded response time scale 702 by "zooming out" via the "+" feature of 702a.

[0094] Referring again to Fig. 7, the graph 700 includes bars 703 which correspond to the response time measurement and data points 704 corresponding to the availability measurement. The bar representations 703 include a colour-coded mid-point 705 corresponding to the 50th percentile of response times of the monitored application per day. In other words, the response time measurements for the day in question span a range (for example from 0.1 seconds to 10 seconds) for the application program's response 211 to service requests 210, and the mid-point 705 represents the response time measurement for which half of the remaining daily measurements were higher (slower) and for which the other half of the remaining daily measurements were lower (faster). In the illustrated example, the bar 703 graphically displays the response times measured for the day in question excluding the fastest 5% and the slowest 5% (i.e., the range spans response times in the 5th to 95th percentile or as otherwise defined by an SLA). The points 704 indicate a single data point for each day corresponding to the percentage of service requests 210 which received a successful service response 211 for the day in question.

[0095] Referring to the specific graph 700, for example, we see that on Feb. 2nd the response time indications correspond to a range of response times from 0.1 seconds up to and presumably beyond 4.0 seconds and a 50th percentile falling around 0.1 seconds. This indicates that most of the response times on Feb. 2, 1998 were clustered in the area of tenths of a second. Likewise on Feb. 2nd the graph 700 indicates that application availability was around 98%.

[0096] In accordance with the foregoing it will be noted that the graph 700 includes "hot points" (704 for example) or "active areas" which are interactive portions of the graph. These hot points are hyperlinked or otherwise interactively connected to related graphs and tables which permit the viewer to dynamically examine a particular point of interest on the graph 700 in greater detail. For example, if a viewer were to perform a mouse click after placing his/her mouse pointer on the background area 708 of the graph (i.e., not on an area showing data for a particular day, which will be subsequently addressed), the viewer would be initiating a request for the data corresponding to a tabular representation of the data illustrated in graph 700 and would correspondingly be presented with the table 800 illustrated in Fig. 8. Thus, it will be understood that the background 708 of the graph 700 is an interactive hot point.

[0097] Table 800 shows a tabular form of the data used to create the graph 700 which is useful in that it provides a more precise reading of data corresponding to a particular day 801. To illustrate we can examine the data for Feb. 2nd. From the table 800 we can now see that the fastest response time 802 was actually 0.071 seconds and significantly, that the slowest response time 803 was 14.062 seconds (the graphical representation 700 is bounded at 4.0 seconds but could be re-scaled using the aforementioned scaling feature 702a as illustrated in table 700a showing response time in the range of 14 seconds for Feb. 2nd. Additionally, the availability data per day 804 and mid-point response time 805 per day is listed in tabular form. A mouse-click on the table 800 dynamically returns the viewer to table 700.

[0098] If a viewer wished to discern at which server and at what time of day the slow 14.062 second response time illustrated on the graph 700, 700a and table 800 occurred, they would place a mouse pointer on the bar 703 or row 801 representing that day, and by clicking on his/her mouse, the AMA GUI will dynamically query the repository 504, retrieve the relevant data, and present the viewer with table 900 shown in Fig. 9.

[0099] Table 900 illustrates availability and response time statistics for each server that is represented on graph 700 for the day of interest (i.e., Feb. 2, 1998). Each server is represented by a hyperlinked or otherwise dynamically related button 901 (hot point) on the left portion of table 900. The table comprises columns representing availability percentages for the day 902, as well as the 5% (903) and 95% (904) and mid-point (905) response times for the day.

[0100] By illustrating the statistical performance of each server in table 900 it is now possible to easily determine which server or servers exhibited poor response time on the day in question. These servers are the machines responsible for causing the graphical representation of poor response time on graph 700 which is a composite of the servers shown in the table 900. Of course, it will be understood that many of the 95% response times for each of the individually listed servers will not be displayed in the 95% response time for the combined server display on graph 700.

[0101] This exclusion, it will be understood, is merely a detailed implementation choice, however, it was included in the preferred embodiment due to the fact that it was discovered that the inclusion of these outlying data points (i.e., response times) skewed the graphical representation of the majority of the monitoring data for the period in question. As such, by excluding these outlying points it was found that the displays more accurately represented the typical performance of the monitored application programs.

[0102] Via reference to table 900, we can see that server D01ML010 exhibited a 5.649 second response time for its 95% response on Feb. 2nd. By clicking on the button 901 corresponding to this server we are presented with graph 1000 as shown in Fig. 10.

[0103] Graph 1000 illustrates, in a similar fashion to that described for graph 700, the performance (i.e., availability and response time) of server D01ML010. Server response time 1002 and availability 1001 is charted for each day 1003 of the month of February 1998.

[0104] In a similar fashion as in graph 700, the bars 1004 represent the response times of the monitored application per day however, in this presentation, the data is depicted only for server D01ML010. As in graph 700, the coloured hash-mark 1005 corresponding to the mid-point of the 5%-95% recorded responses for the day. The data points 1006 correspond to the availability readings for the application program per day. Since we know that server D01ML010 had a 95% response time of 5.649 seconds, the reading exceeds the upper bounds of graph 1000, however in graph 1000a in Fig. 10a the response time boundary has been expanded to show the 5.649 response time for this server on Feb. 2, 1998.

[0105] If a viewer wished to examine this data in tabular form, a mouse-click (or other such viewer-based interaction) on the background area would reveal a tabular representation of this data identical to table 800, however, the table would only represent the data in table 1000 which corresponds to server D01ML010. Since this table is virtually identical to table 800 it is not necessary to illustrate this representation, however it is significant to note that each graphical representation is dynamically linked to a table which more specifically illustrates the data underlying the graphical representation.

[0106] Referring again to Fig. 10, if a viewer wished to examine a graph of the hourly performance of server D01ML010 for Feb. 2, 1998, s/he would place a pointer on the bar 1004 corresponding to that date and by mouse-clicking would prompt the dynamic generation of graph 1100 shown in Fig. 11.

[0107] Graph 1100 is identical in format to graph 700 and graph 1000, however graph 1100 depicts availability 1101 and response time 1102 for server D01ML010 on an hourly basis 1103 on the date Feb. 2, 1998. The same indicia 1104, 1105 and 1106 are used for the response times and availability indicators, but on this graph 1100 the bars now show maximum, mid-point and the minimum values.

[0108] From this graphical illustration 1100 the viewer can determine at which time(s) of day the server exhibited poor response-time performance. Accordingly, by examining graph 1100 for Feb 2nd for server D01ML010 we can see that during the hours of 10 A.M., 11 A.M. and 3 P.M. the response times appeared to be exceptionally slow. However, since graph 1100 is bounded for response times at 4.0 seconds the viewer would push the "+" button associated with the response time scale 1102a and would be presented with graph 1100a in Fig. 11a wherein it can be seen that the response time for the hour of 10 A.M. was over 27 seconds, the response time for the hour of 11 A.M. was around 25 seconds and for the hour of 3 P.M. the response time was approximately 6 seconds.

[0109] If the viewer wished to see the details for this graph 1100 he/she would click anywhere on the graph and be presented with table 1200 as shown in Fig. 12 which in turn illustrates a tabular presentation of the performance data comprising graph 1100. From this table it can be seen that the response time for D01ML010 running the monitored Lotus Notes (AIX-based) mail application was actually a worst-case 73.281 seconds during the 10 o'clock A.M. hour.

[0110] Fig. 13 represents an overview of the display and report generation facilities provided by the present invention. In general, the invention provides a mechanism for interacting with tabular or graphic display representations of data from repository 504. In step 1301 the user requests the retrieval and display of a first set of data (i.e., the availability and response time data for Lotus Notes (AIX) mail application for the month of February for all servers serving the Poughkeepsie, NY office location (see Fig. 7)). In step 1302 the GUI 605 forwards the request to the bridge daemon 606 which in turn queries the repository 504 and then returns the retrieved data to the GUI which then displays the requested data in step 1303. The user interacts with designated hot points on the display in step 1304 which in turn initiates a request for a second set of data related to the data shown and interacted with on the first display. Next in step 1305 the GUI 605 again passes the request to the AMA bridge daemon 606 which queries the repository 504 for the requested related data, the data is provided via the bridge to the GUI 605 1305 which then builds the display of the second set of related data based upon the data returned from the repository in step 1306.

[0111] In Fig. 14 which can be read in conjunction with Fig. 13 it can be seen that the interrelation of the successive displays described above can be viewed as an inverted tree graph 1400. Starting at block 1401 a viewer requests via a menu (or otherwise) on his computer system the display of a data set including application program monitoring data as indicated in block 1402. It will be noted that block 1402 includes both a graphical representation of the monitoring data set illustratively represented by bar graph 1402a and a tabular representation of the monitoring data set shown

as table 1402b. The table and graph are dynamically linked to one another by link 1402c such that a viewer may jump between a tabular display of the data set and a graphical display of the data set by clicking on (or otherwise interfacing with) an active display area on the viewer's computer system.

[0112] Within the data set depicted in table 1402b or graph 1402a are data elements represented as A and B. Each of these data elements in both the table and graph may be implemented as the aforementioned active areas or hot-spot such that they may be indicated via a mouse click or other viewer interface technique. In response to the viewer's indication, the dynamic linking activities as depicted in Fig. 13 are implemented to provide the display of a second data set including associated data elements. For example, if the viewer were to indicate data element A within table 1402b or graph 1402a the sequence illustrated in Fig. 13 would be initiated for dynamic link 1403a to subsequently display the table and/or graphic display represented by block 1404.

[0113] In block 1404 a graphic representation 1404a and a tabular representation 1404b are shown for the displayed second data set. The second data set includes data elements A_1 , A_2 and A_3 which are associated to the data element A in the data set shown in graph 1402a or table 1042b. Each of these associated elements in a preferred embodiment represents a component of the data element A. For example, in an embodiment wherein the data element A represents the range of response times recorded for a given day for a Lotus Notes application program response to three different probes (i.e., probe 1, probe2 and probe 3), the data set shown in 1404a and 1404b includes the range of response time recorded at each probe for that given day, such that A_1 represents response data for probe 1 on the given day, A_2 represents the responses received at probe 2, and A_3 represents the responses at probe 3 for the given day. It will of course be understood that graph 1404a and table 1404b are dynamically linked to each other via 1404c and can be toggled in a manner similar to that described for the data sets 1402a and 1402b. via link 1402c. It will further be understood that each data element shown in data set 1402 (i.e., A and B) may be linked 1403a and 1403b to further representations as has been described for data element A.

[0114] Each of the data elements in the data set shown in 1404a and 1404b may be further dynamically linked via links 1406 to further data sets (not shown) which in turn include data elements which represent components of the data elements in the data set 1404a and 1404b (i.e., the data elements A_1 , A_2 and A_3).

[0115] It will be appreciated from the foregoing exemplary embodiment, that the AMA GUI in combination with the bridge 606 and repository 504 enable an end-user to quickly and efficiently analyze voluminous data records. In the foregoing simplified example a viewer was permitted to "drill-down" from an overview of server performance to localized a response time aberration to a specific server at a specific time of day.

[0116] It will be further appreciated that the foregoing techniques may be coupled with available data corresponding to server specific performance, such as data provided by IBM Netfinity or other such products to further refine the problem determination process. The granularity of this determination process is limited only by the amount and types of network performance measurements gathered for a given network and by the limitations of data mining and analysis tools for correlating this collected information.

[0117] A further inventive feature of the present invention entails the techniques utilized for providing alerts to service personnel upon the determination that any of the defined performance criteria have been violated.

[0118] As previously described, the AMA probe 201 may be provided with probe configuration information 202, including threshold information such as maximum response time or minimum application availability. These criteria are monitored by the AMA probe code 303 and when a violation is detected an alert signal 308 is generated by the probe for informing an alerting mechanism 205 of the violation. The alerting mechanism 205 may in turn function by signalling the violation to a support person, or otherwise actuating a problem remediation response.

[0119] Fig. 15 illustrates an exemplary implementation of this alerting sequence 1500 in more detail. When a threshold defined as part of the probe configuration information 302 has been violated as determined by the AMA probe code 303, an alert signal 308 is sent to the alerting mechanism 205 which is illustrated in the preferred embodiment as a server computer 1501 running the IBM NetView (R) software application program (hereinafter the NetView server). In the preferred embodiment the threshold violation causes the AMA probe code to generate a software indication known as a trap 308. This trap indication is defined in accordance with the transmission control protocol/internet protocol (TCP/IP), simple network messaging protocol (SNMP) configuration for client-server network based communication. This trap indication includes information about the server computer 202 and application program 203 therein which is in violation, and the type of violation that has been recorded. The aforementioned software trap is well-known to those having skill in the art and as such needs no detailed explanation. It will be further understood that while the present invention implements a software trap to signal the occurrence of a violation, other mechanisms such as e-mail, or paging, pop-up screens or other notification methodologies may be utilized as well.

[0120] Once the trap 308 is received at the NetView server 1501 it is appended to a repository trapd.log 1502 within the server. The repository 1502 is in turn scanned at pre-defined intervals by another server 1503 (illustrated as the IBM Global Services Coordinator (GSC) server).

[0121] The GSC server 1503 compares any new entries in the trapd.log 1502 with a set of tables 1504 within the GSC server. These tables 1504 include information related to the application type, server location violation type and

the service person who is to be informed of the violation, how to inform that person (i.e., e-mail, paging, etc..) which may be based upon such factors as the criticality of the violation and the time of day.

[0122] If the GSC server is able to match the server, application and violation type from trapd.log 1502 with an entry in its tables 1504, an alert signal 1505 is generated in accordance with the specified modality 1506 (i.e., pager, e-mail etc..) to the indicated party.

[0123] If the AMA probe code 303 subsequently determines that a violated threshold has return to an acceptable level, further facilities may be provided for cancelling the alert. In such an embodiment, the cancellation would follow the same sequence as the original violation indication wherein a new software trap 308 would be generated, this time indicating that the violation has been resolved. The trap would be stored in trapd.log 1502 and scanned by the GSC server 1503 whose tables 1504 would indicate that the alert should be cancelled which in turn would either stop the alert signal 1505 prior to transmission 1506 or alternatively transmit 1506 a message 1505 cancelling the prior alert.

Claims

1. A method for monitoring and recording, at a client computer, information related to the performance of application services by an application program, the method being for use in a computer network having a server computer coupled to a client computer, the server computer providing application services to the client computer over the network via an application program on the server computer, the method comprising the steps of:

A. establishing a set of parameters for use by the client computer for recording the performance of said application services by said application program, wherein said set of parameters may include the definition of an interval for iteratively assessing the performance of the application program;

B. sending a service request from the client computer to the server computer, said service request requesting the performance of said application services by the application program based upon the established set of parameters;

C. receiving a service response from the server computer at the client computer and generating a transaction record at the client computer based upon the received service response, said generated transaction record including the information related to the performance of said application services by said application program; and

D. repeating steps B - C as required in accordance with the defined interval of iteration.

2. A method as claimed in claim 1 wherein the set of parameters further includes the information required to send the service requests to the application program at the server computer.

3. A method as claimed in claim 1 wherein the set of parameters includes the frequency at which the service requests are to be sent.

4. A method as claimed in claim 1 wherein the generated transaction record includes an indication of the duration of the period from the sending of the service request to the receiving of the service response.

5. A method as claimed in claim 1 wherein the transaction record includes an indication of whether the service response was successful or unsuccessful and wherein the set of parameters include a maximum duration for the period from the sending of the service request to the receiving of the service response.

6. A method as claimed in claim 5 wherein if said maximum duration is exceeded the service request is re-sent from the client computer to the server computer.

7. A method as claimed in claim 6 wherein the operation of re-sending the service request may be repeated if the maximum duration is exceeded after the first re-sending operation and wherein the set of parameters further include a limit on said re-sending operation said limit based upon the duration of the repeated re-sending operations or on the number of times the re-sending operation is repeated.

8. A method as claimed in claim 7 wherein if said limit is exceeded the service response is recorded as unsuccessful.

9. A method as claimed in claim 5 wherein if the application program fails to provide application services in response to the service request, the service response is recorded as unsuccessful.

10. A method as claimed in claim 1 wherein the set of parameters further include a schedule corresponding to times when the application program will be available to respond to the service requests.

11. A method as claimed in claim 10 wherein the schedule is used to enable and disable the sending of the service requests or wherein the generated transaction record includes an indication of whether the application program was available in accordance with the schedule.

12. A method as claimed in claim 1 wherein the set of parameters include a set of thresholds related to the performance of the application services and wherein the method further includes the steps of:

analyzing the generated transaction record to determine if any of the thresholds in the set of thresholds has been exceeded by the response to the transaction request; and
sending an alert signal to an alerting mechanism if it is determined that any of the thresholds in the set of thresholds has been exceeded.

13. A method as claimed in claim 1 wherein the set of parameters for use by the client computer for recording the performance of said application services by said application program are provided via a user interface from a user of the client computer to the client computer and wherein information related to the received service responses and generated transaction records are provided from the client computer to the user interface for viewing by said user.

14. An apparatus for monitoring and recording at a client computer, information related to the performance of application services by an application program, the apparatus being for use in a computer network having a server computer coupled to a client computer, the server computer providing application services to the client computer over the network via an application program on the server computer, the apparatus comprising:

means for establishing a set of parameters for use by the client computer for recording the performance of said application services by said application program, wherein said set of parameters may include the definition of an interval for iteratively assessing the performance of the application program;

means for sending a service request from the client computer to the server computer, said service request requesting the performance of said application services by the application program based upon the established set of parameters;

means for receiving a service response from the server computer at the client computer and means for generating a transaction record at the client computer based upon the received service response, said generated transaction record including the information related to the performance of said application services by said application program.

15. A program storage device readable by a digital processing apparatus and tangibly embodying a program of instructions executable by the digital processing apparatus to perform method steps for monitoring and recording at said client computer, information related to the performance of said application services by said application program, the program storage device being for use in a computer network having a server computer coupled to a client computer, the server computer providing application services to the client computer over the network via an application program on the server computer, the method comprising the steps of:

A. establishing a set of parameters for use by the client computer for recording the performance of said application services by said application program, wherein said set of parameters may include the definition of an interval for iteratively assessing the performance of the application program;

B. sending a service request from the client computer to the server computer, said service request requesting the performance of said application services by the application program based upon the established set of parameters;

C. receiving a service response from the server computer at the client computer and generating a transaction

EP 0 957 432 A2

record at the client computer based upon the received service response, said generated transaction record including the information related to the performance of said application services by said application program; and

5 D. repeating steps B - C as required in accordance with the defined interval of iteration.

10

15

20

25

30

35

40

45

50

55

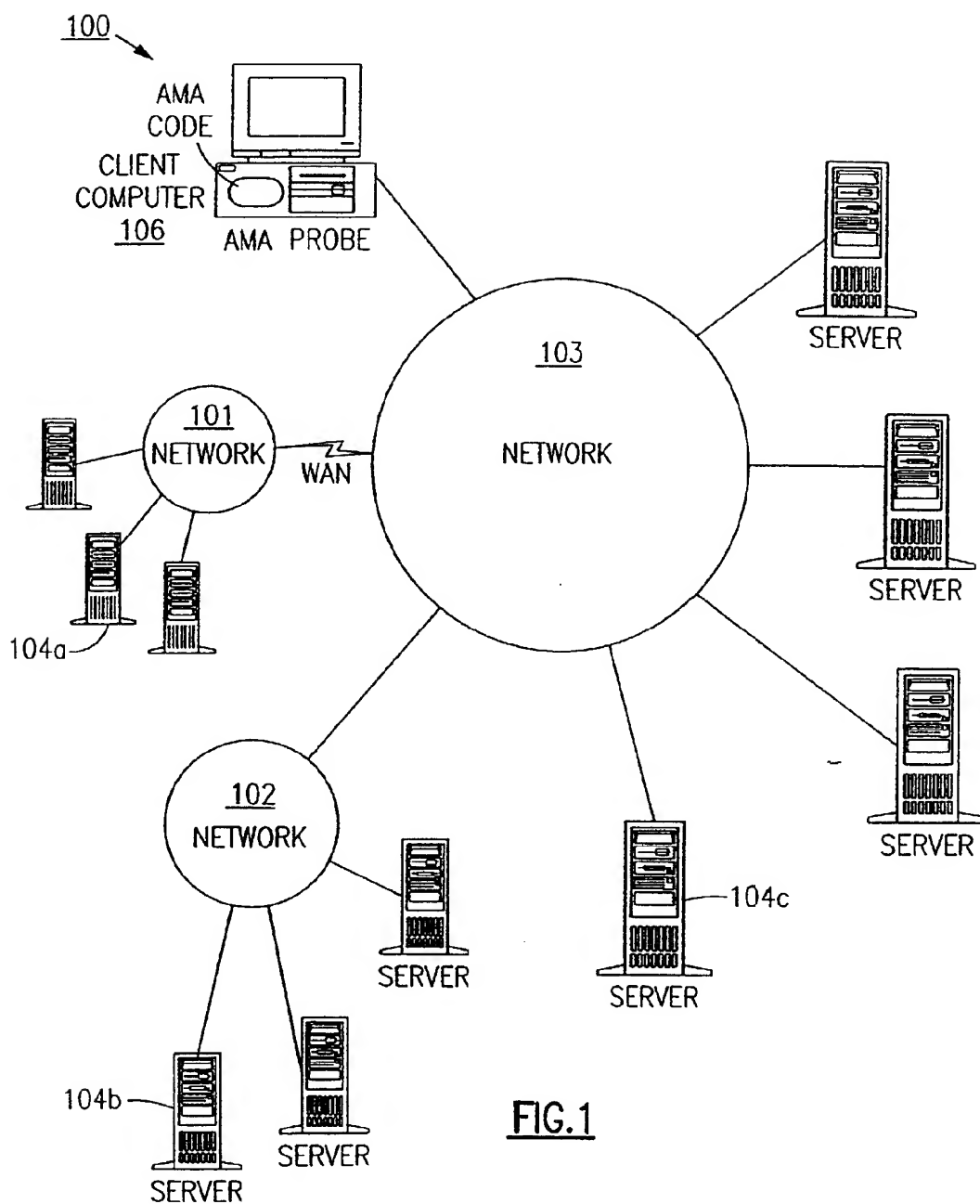


FIG.1

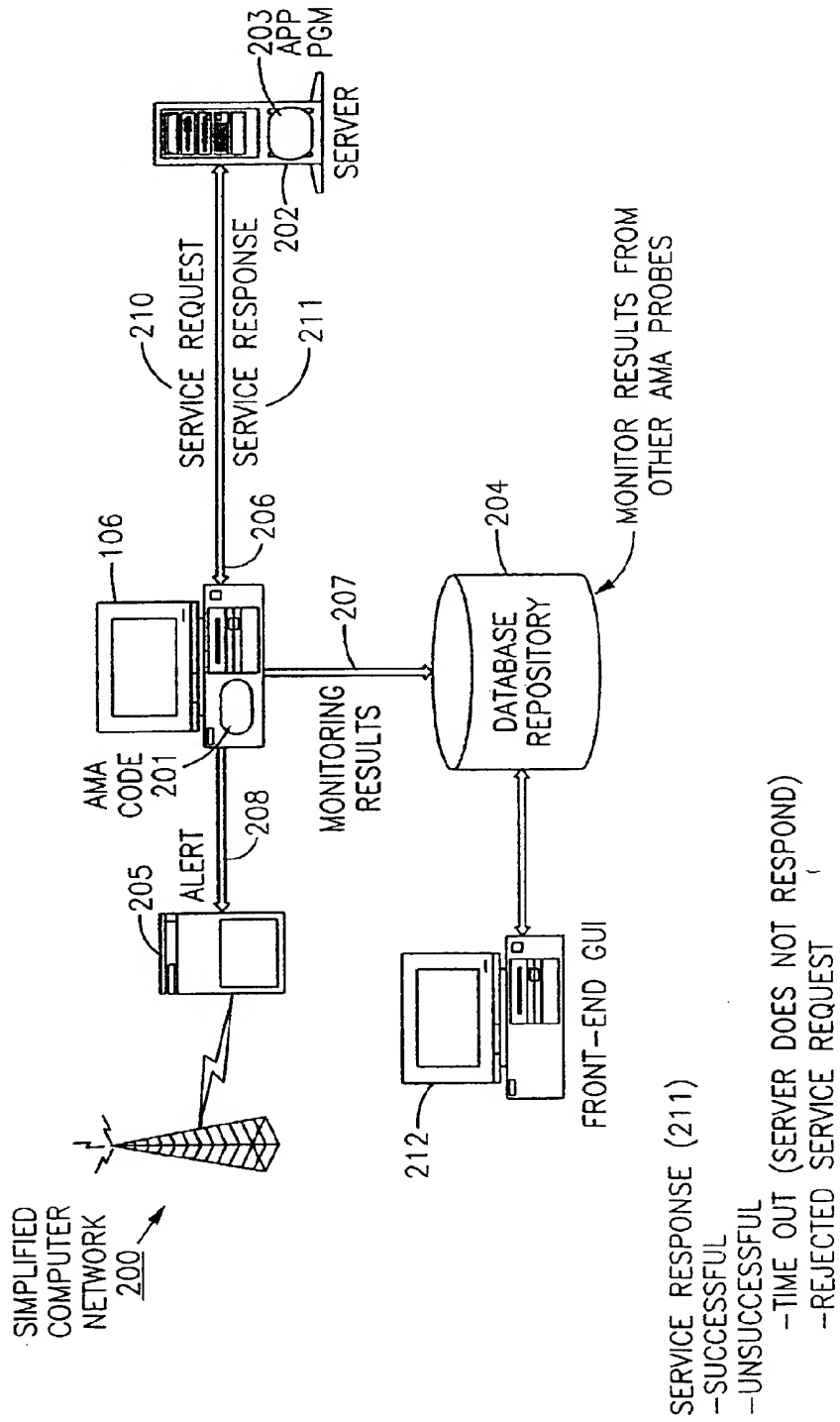


FIG. 2

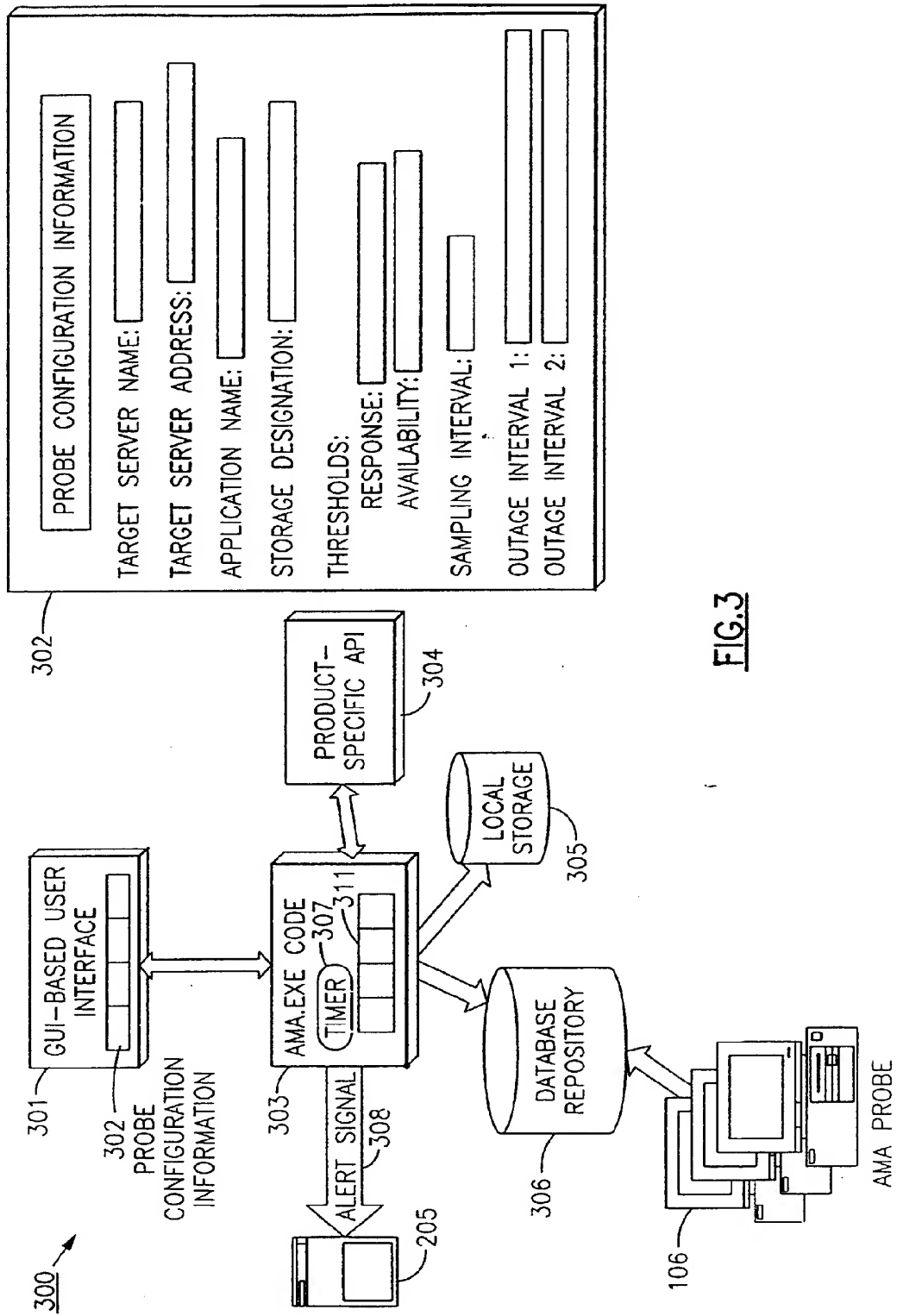
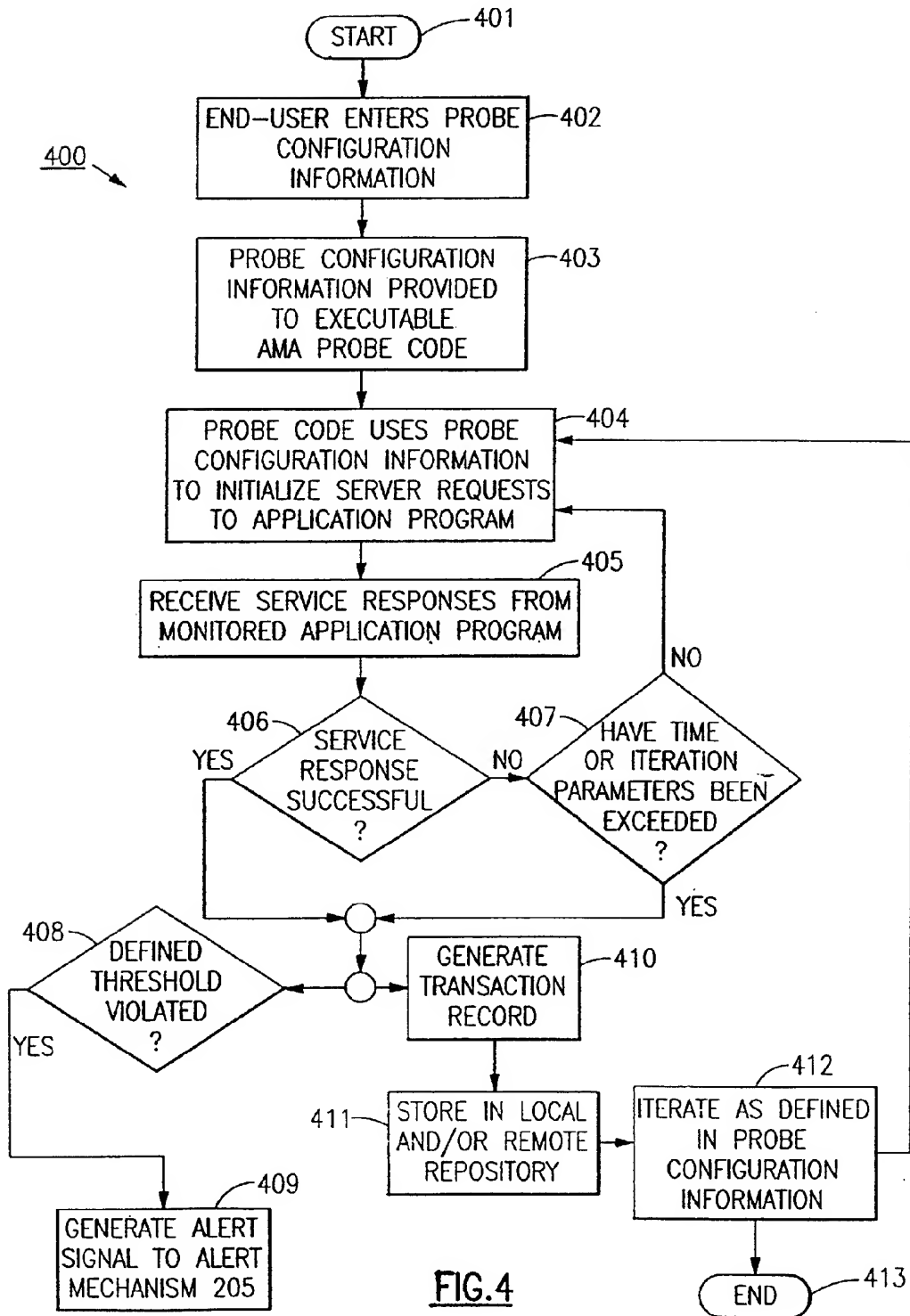


FIG. 3



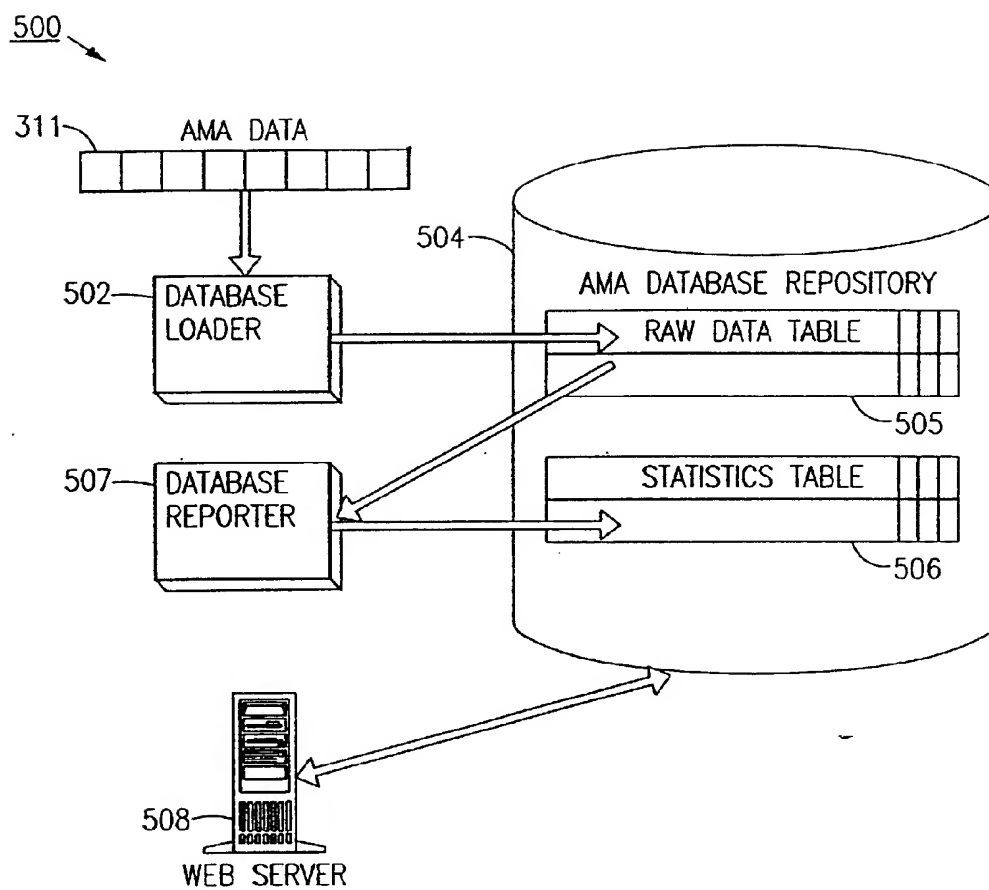


FIG.5

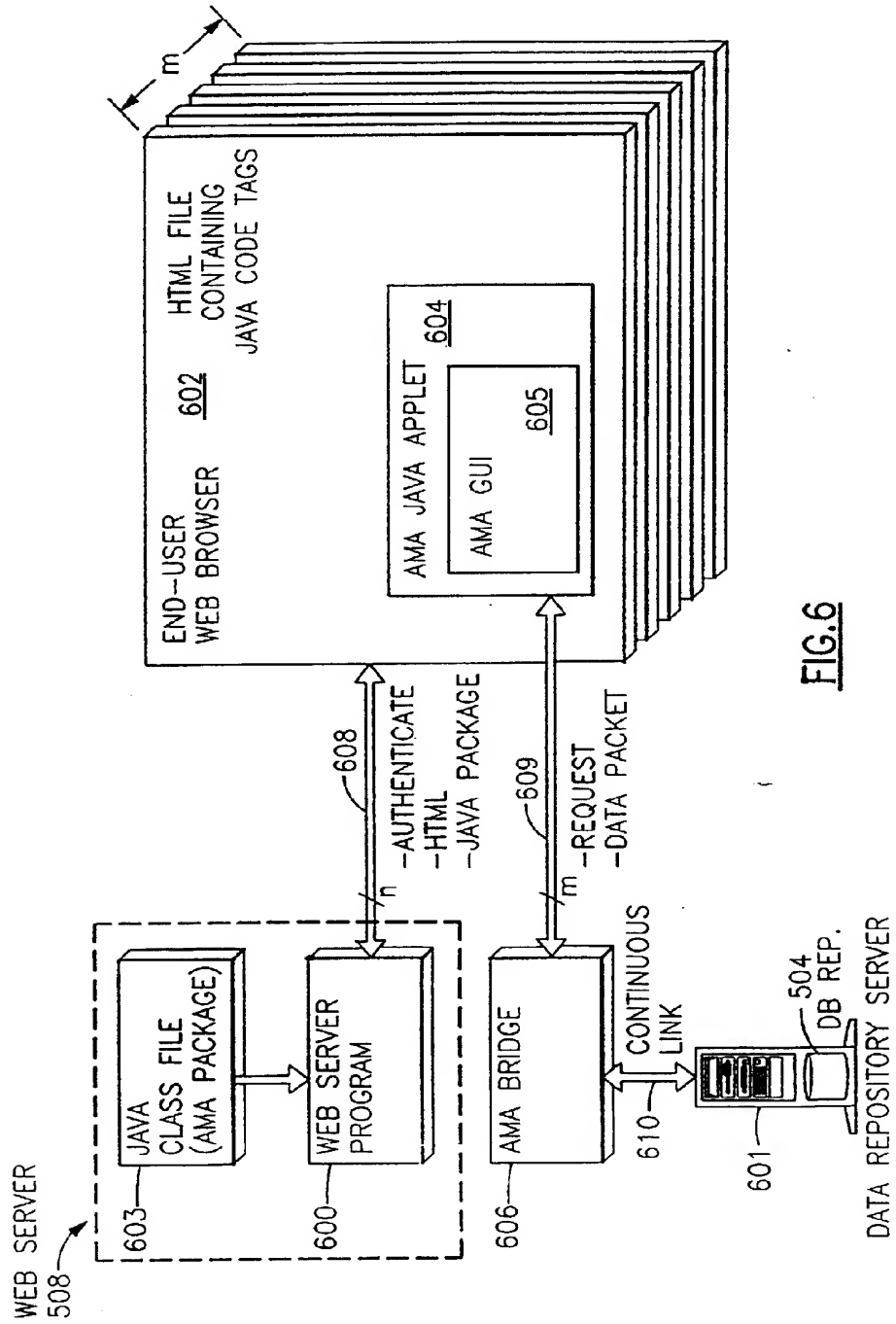


FIG. 6

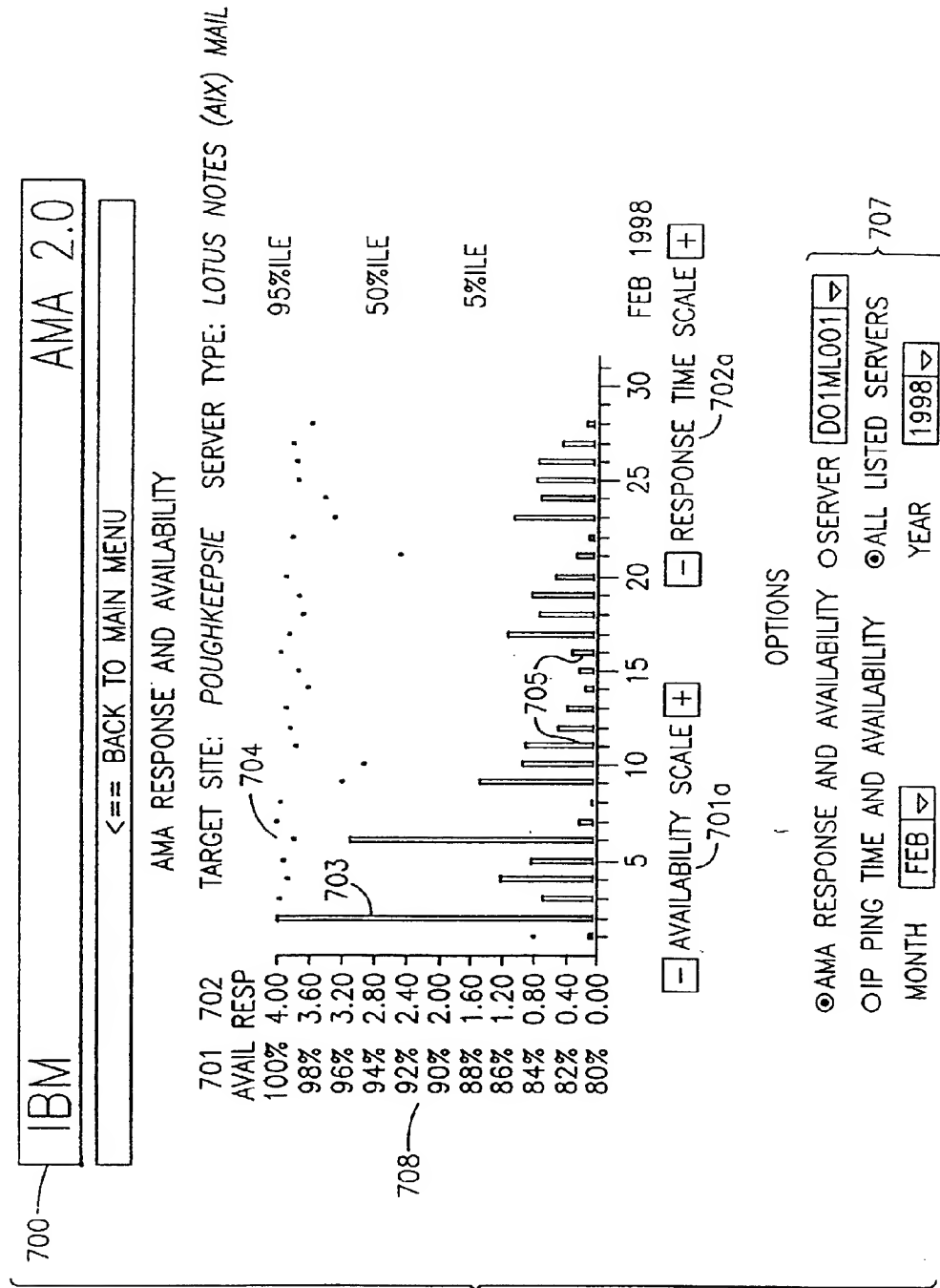


FIG.7

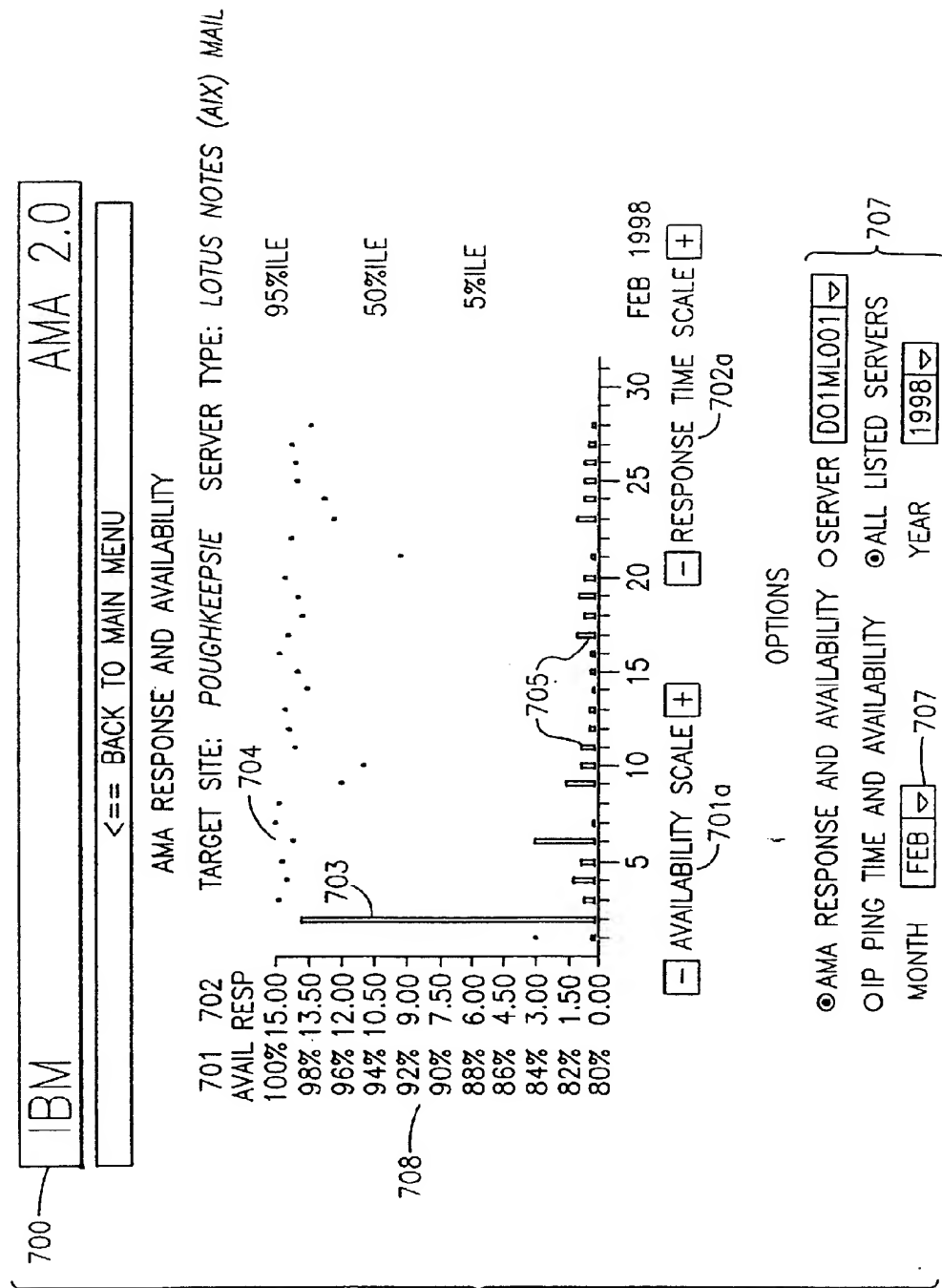


FIG. 7a

800 — IBM — AMA 2.0

<== BACK TO MAIN MENU

AMA RESPONSE AND AVAILABILITY

TARGET SITE: POUGHKEEPSIE SERVER TYPE: LOTUS NOTES (AIX) MAIL

FEB 1998 — 804 — 802 — 805 — 803

DAY	AVAIL	5%ILE	50%ILE	95%ILE
1	83.7%	0.055	0.070	0.094
2	98.2%	0.071	0.110	14.062
3	99.9%	0.079	0.118	0.580
4	99.4%	0.071	0.125	1.227
5	99.6%	0.078	0.140	0.836
6	99.0%	0.078	0.148	3.133
7	100.0%	0.070	0.109	0.242
8	99.8%	0.055	0.086	0.117
9	96.1%	0.078	0.172	1.515
10	94.8%	0.070	0.125	0.954
11	98.9%	0.071	0.125	0.906
12	99.3%	0.078	0.125	0.492
13	99.6%	0.063	0.109	0.399
14	98.2%	0.063	0.094	0.157
15	98.8%	0.062	0.102	0.235

801 —

	99.8%	0.070	0.102	0.328
16	99.8%	0.070	0.117	1.179
17	99.3%	0.070	0.117	0.742
18	98.5%	0.070	0.117	0.843
19	98.7%	0.078	0.148	0.531
20	99.6%	0.070	0.117	0.274
21	92.4%	0.071	0.141	0.133
22	99.2%	0.062	0.078	1.093
23	96.5%	0.070	0.102	0.750
24	97.1%	0.070	0.117	0.804
25	98.8%	0.071	0.117	0.774
26	98.9%	0.070	0.117	0.492
27	99.2%	0.070	0.117	0.164
28	98.1%	0.070	0.102	

OPTIONS

☒ AMA RESPONSE AND AVAILABILITY
 ☐ SERVER D01ML001

☐ PING TIME AND AVAILABILITY
 ☒ ALL LISTED SERVERS

MONTH FEB YEAR 1998

FIG.8

900 — IBM — AMA 2.0

<== BACK TO MAIN MENU

AMA RESPONSE AND AVAILABILITY

TARGET SITE: POUCHKEEPSIE SERVER TYPE: LOTUS NOTES (AIX) MAIL

FEB 2,1998 — 902 — 903 — 904 — 905 — 95%ILE

SERVER — 901 AVAIL 5%ILE 50%ILE 95%ILE

D01ML001	97.2%	0.070	0.101	20.563
D01ML002	99.3%	0.078	0.141	3.313
D01ML003	97.2%	0.078	0.125	29.875
D01ML004	99.4%	0.086	0.109	6.907
D01ML005	96.8%	0.063	0.109	38.594
D01ML006	99.4%	0.078	0.110	11.633
D01ML007	98.6%	0.071	0.110	10.532
D01ML008	98.7%	0.094	0.125	48.242
D01ML009	99.4%	0.101	0.148	11.398
D01ML010	93.6%	0.070	0.101	5.649

OK

FIG.9

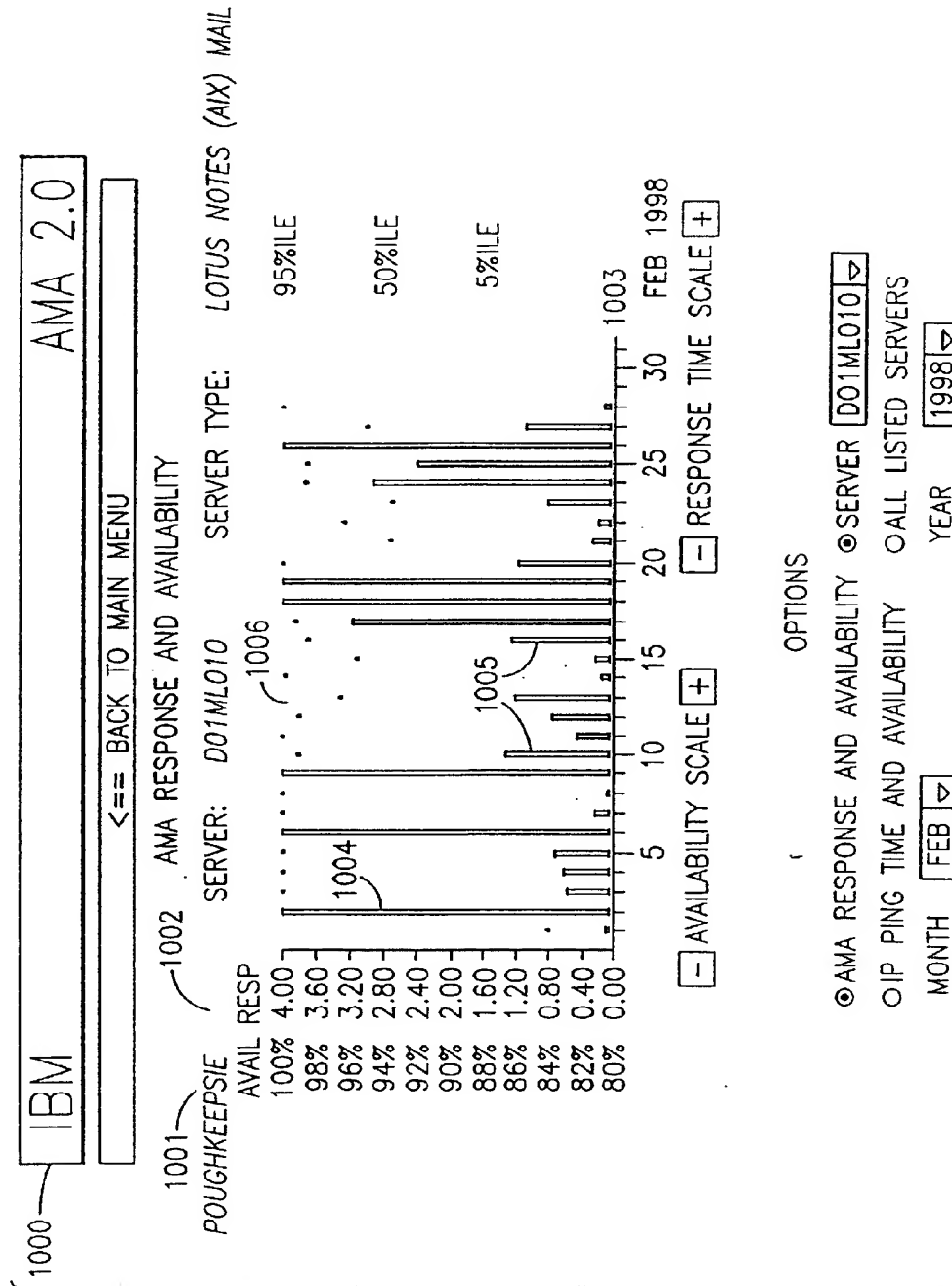
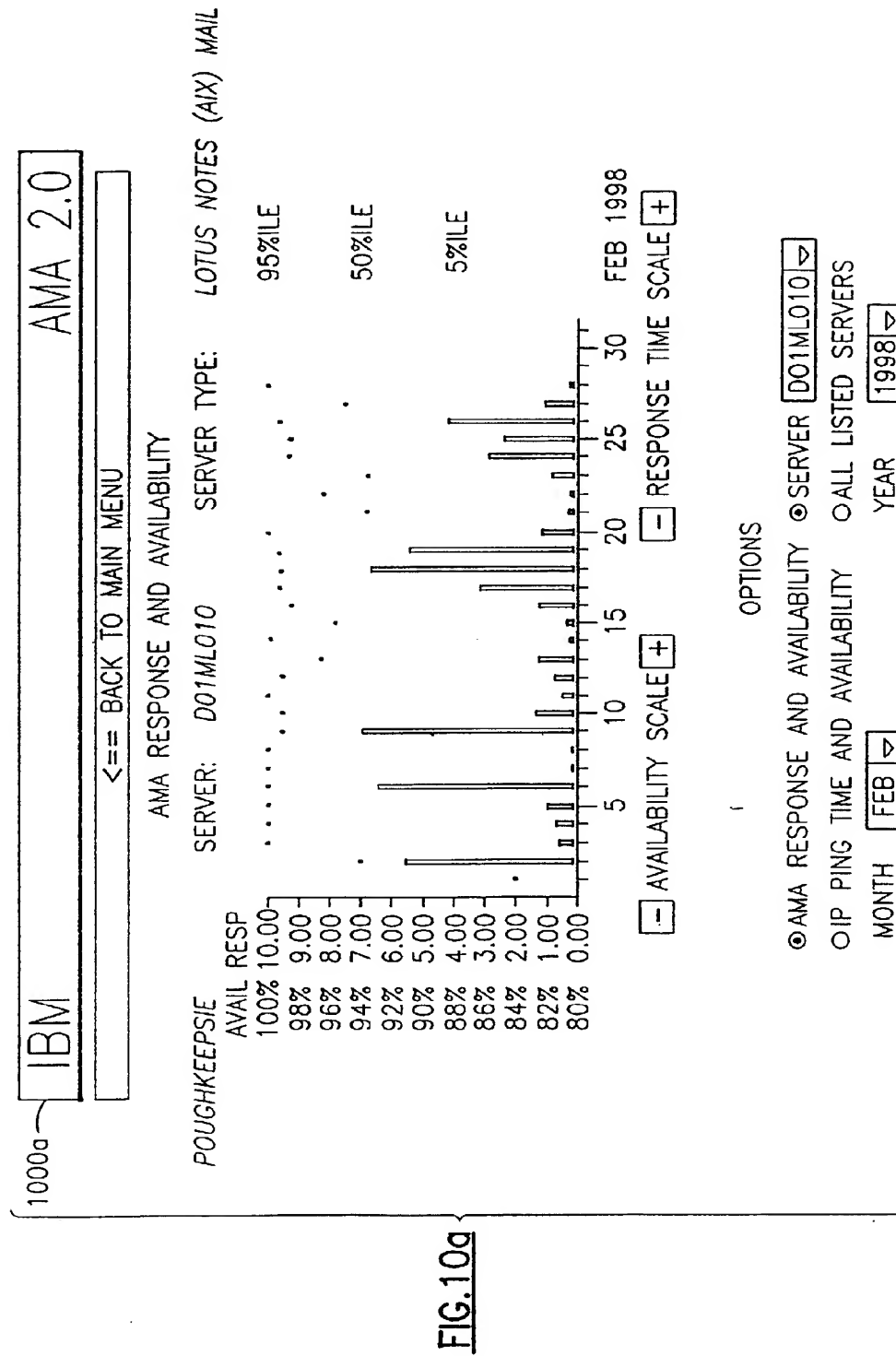


FIG.10



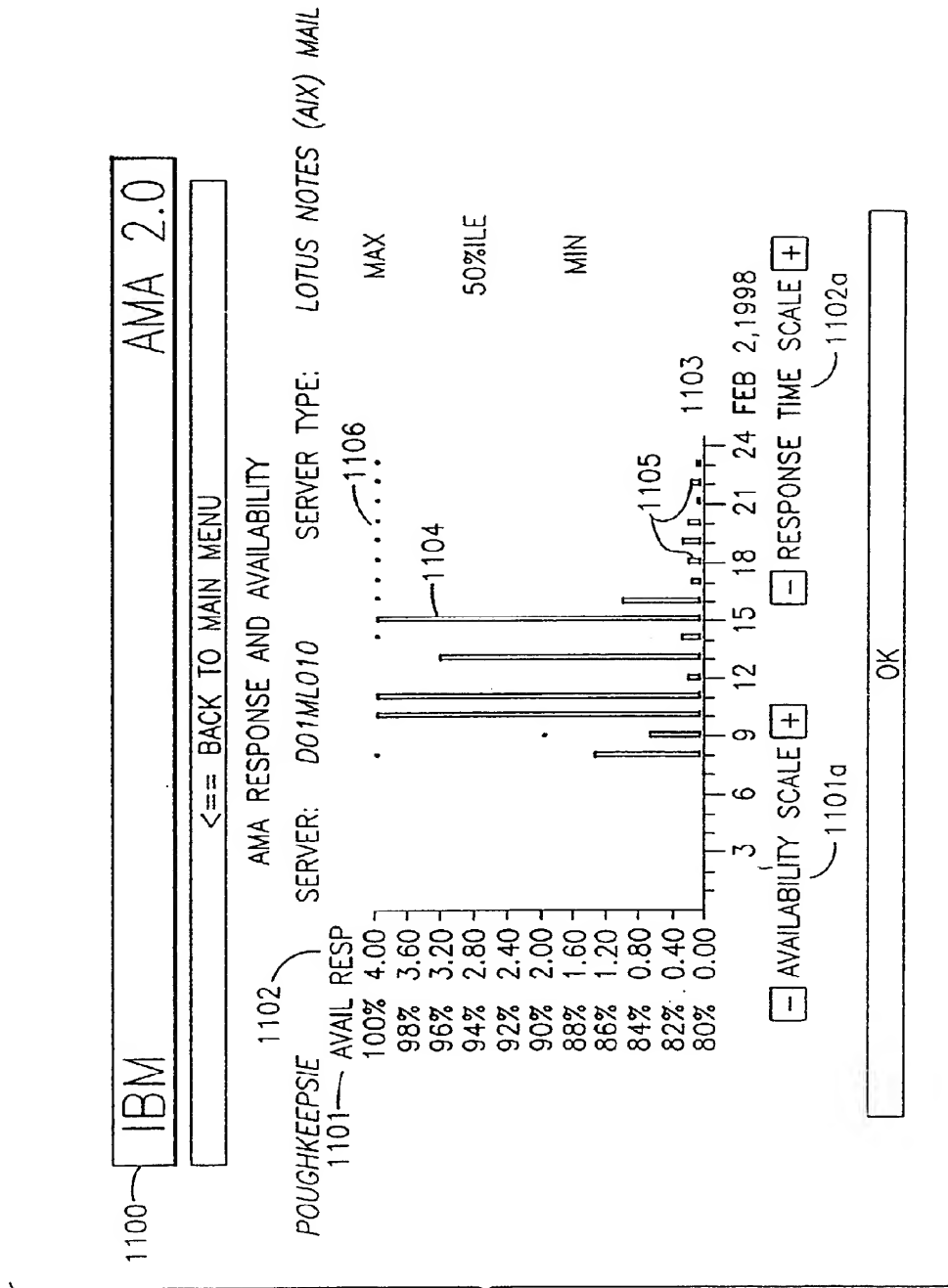


FIG.11

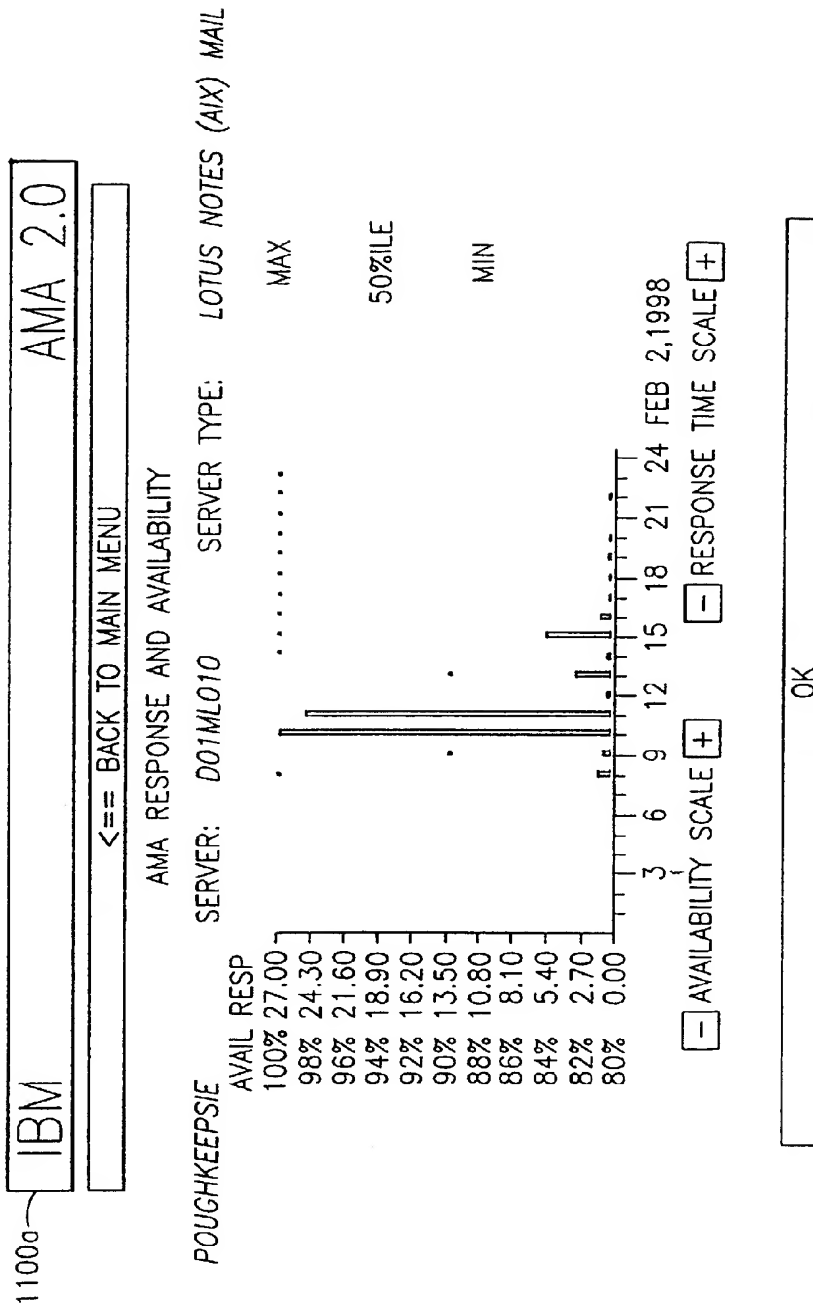


FIG.11a

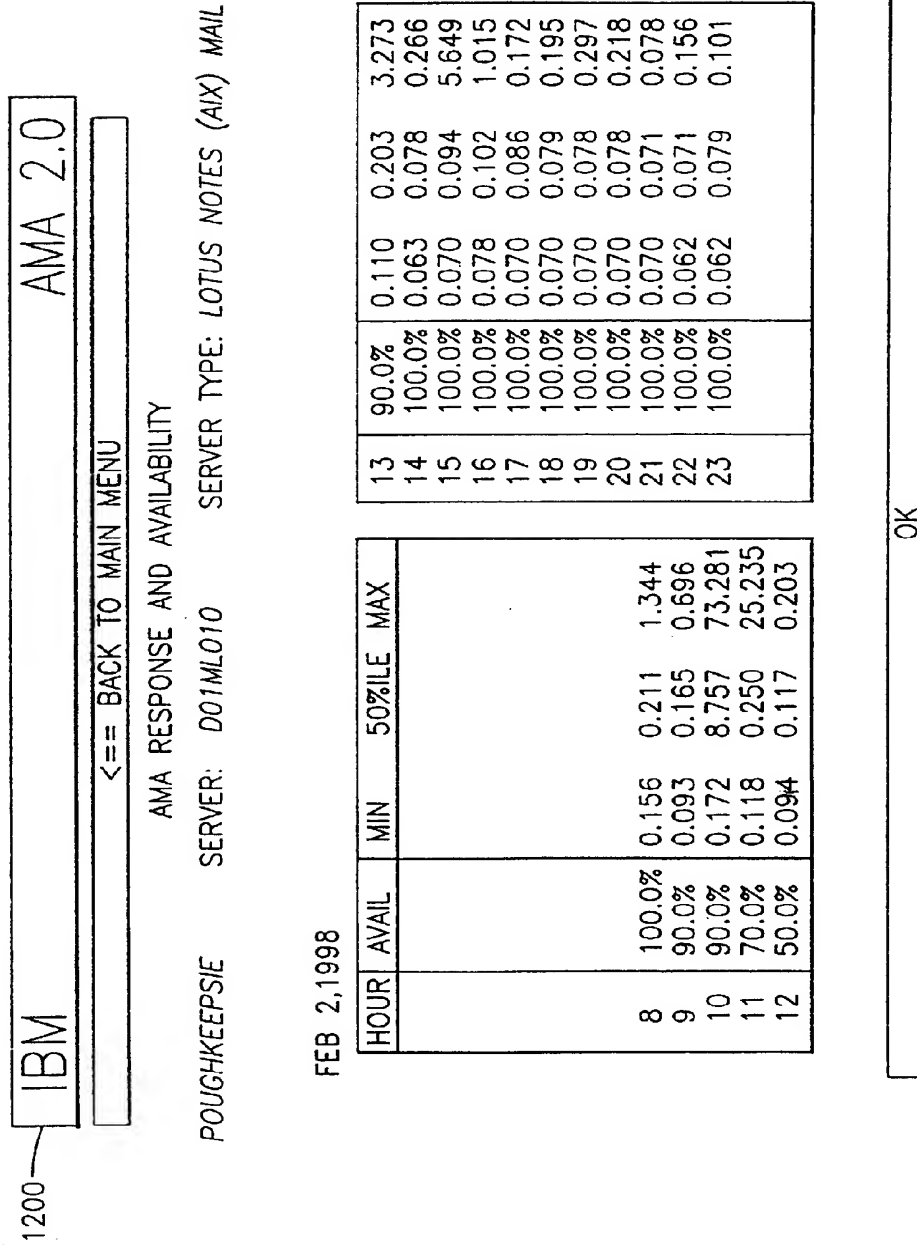
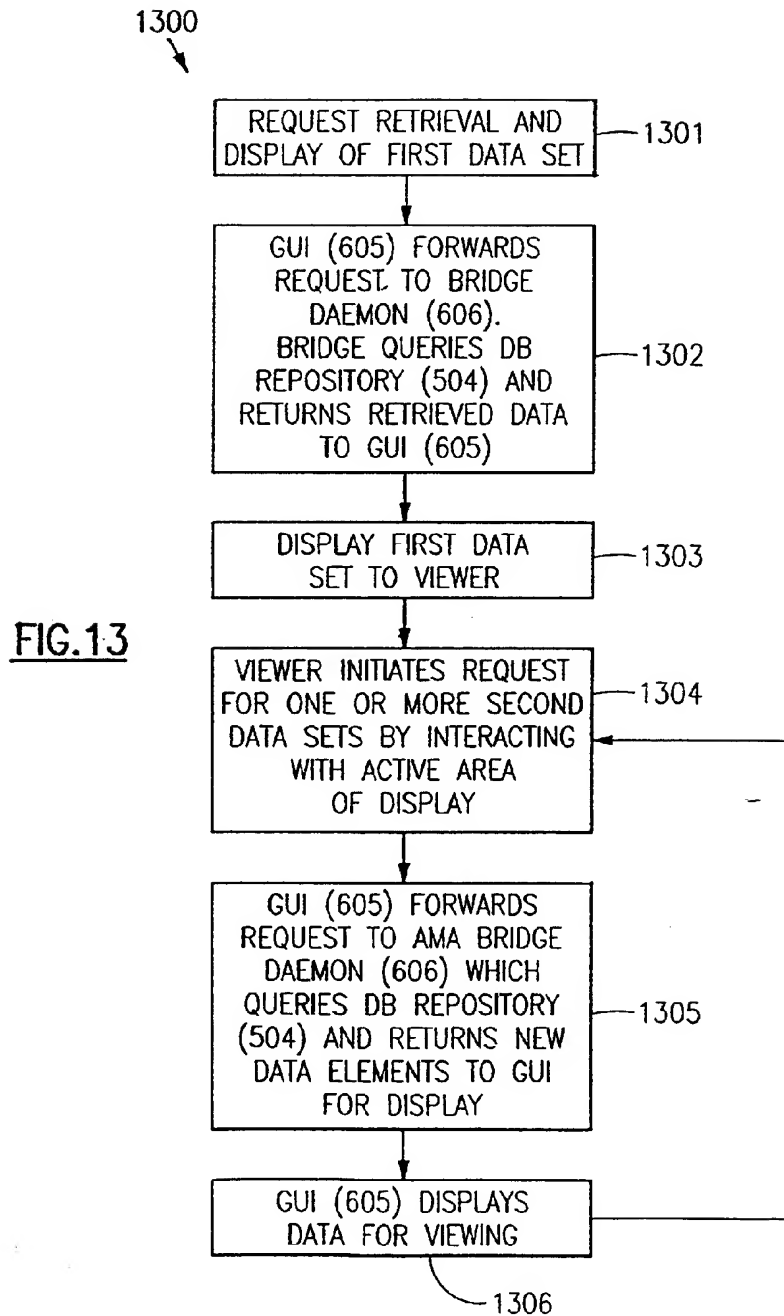


FIG.12



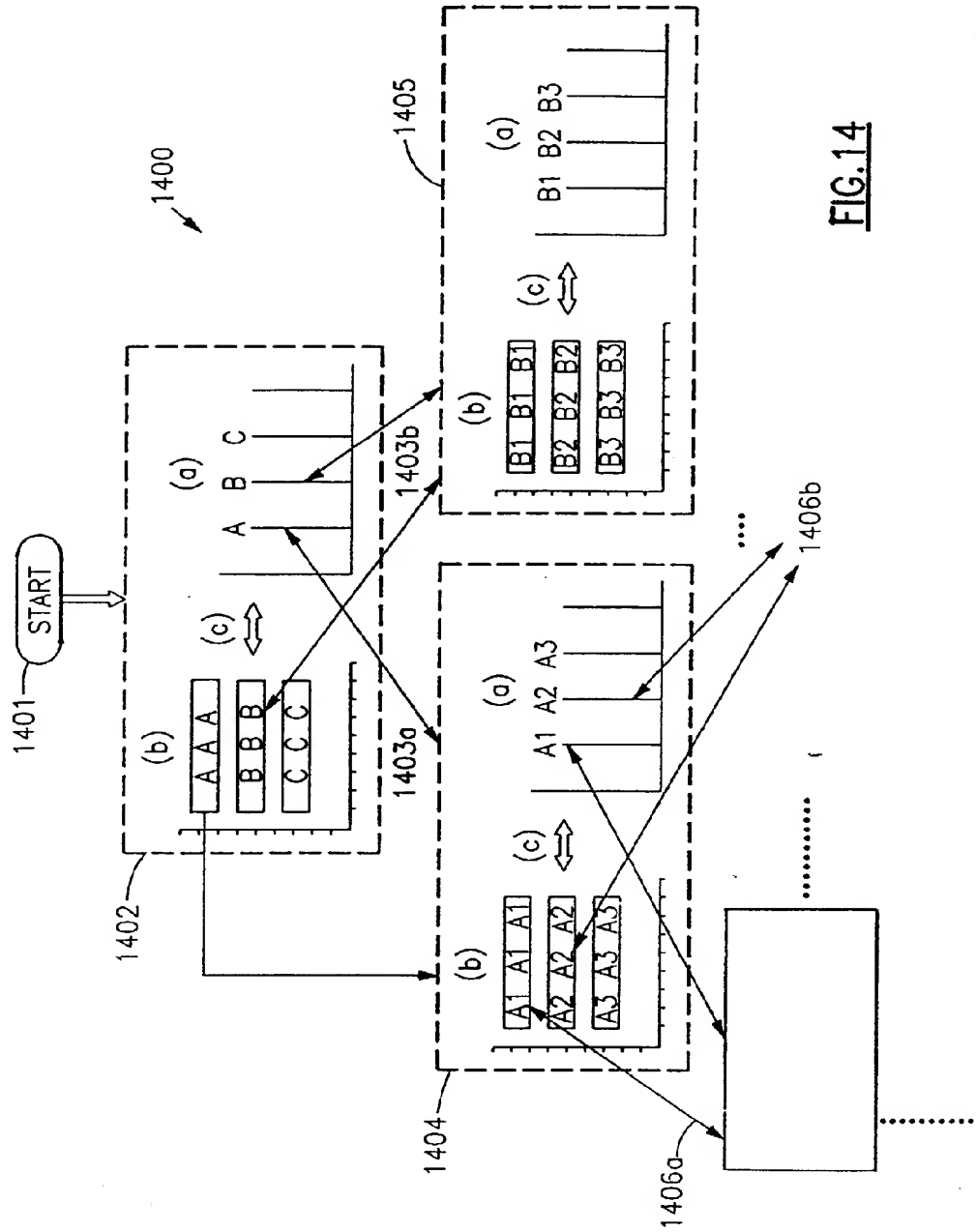


FIG. 14

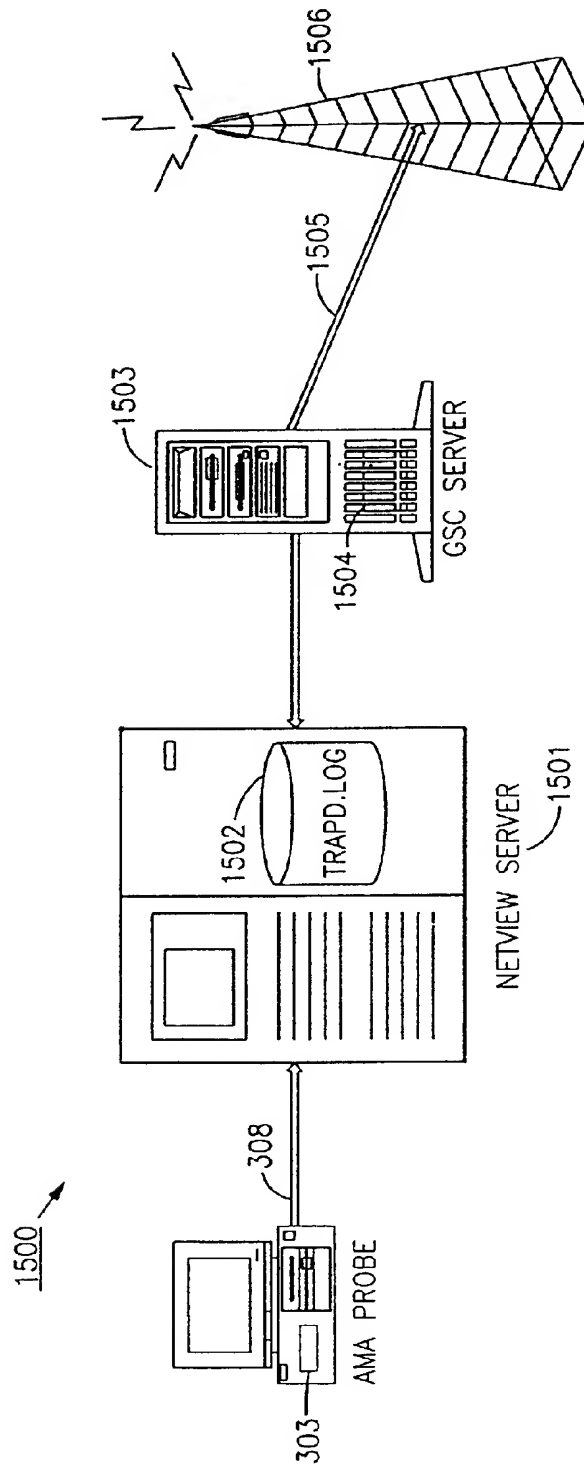


FIG. 15